Quantified Differential Dynamic Logic for Distributed Hybrid Systems

André Platzer

May 2010
CMU-CS-10-126

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work supported by the National Science Foundation under Grant Nos. CNS-0926181 and CNS-0931985, by the NASA grant NNG-05GF84H, and by the ONR award N00014-10-1-0188. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution or government.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **MAY 2010**	2. REPORT TYPE	3. DATES COVERED **00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE **Quantified Differential Dynamic Logic for Distributed Hybrid Systems**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We address a fundamental mismatch between the combinations of dynamics that occur in complex physical systems and the limited kinds of dynamics supported in analysis. Modern applications combine communication, computation, and control. They may even form dynamic networks where neither structure nor dimension stay the same while the system follows mixed discrete and continuous dynamics. We provide the logical foundations for closing this analytic gap. We develop a system model for distributed hybrid systems that combines quantified differential equations with quantified assignments and dynamic dimensionality-changes. We introduce a dynamic logic for verifying distributed hybrid systems and present a proof calculus for it. We prove that this calculus is a sound and complete axiomatization of the behavior of distributed hybrid systems relative to quantified differential equations. In our calculus we have proven collision freedom in distributed car control even when new cars may appear dynamically on the road.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**32**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Keywords: Dynamic logic, Distributed hybrid systems, Axiomatization, Theorem proving, Quantified differential equations

Abstract

We address a fundamental mismatch between the combinations of dynamics that occur in complex physical systems and the limited kinds of dynamics supported in analysis. Modern applications combine communication, computation, and control. They may even form dynamic networks, where neither structure nor dimension stay the same while the system follows mixed discrete and continuous dynamics.

We provide the logical foundations for closing this analytic gap. We develop a system model for distributed hybrid systems that combines quantified differential equations with quantified assignments and dynamic dimensionality-changes. We introduce a dynamic logic for verifying distributed hybrid systems and present a proof calculus for it. We prove that this calculus is a sound and complete axiomatization of the behavior of distributed hybrid systems relative to quantified differential equations. In our calculus we have proven collision freedom in distributed car control even when new cars may appear dynamically on the road.

1 Introduction

Many safety-critical computers are embedded in cyber-physical systems like cars [13] or aircraft [7]. How do we know that their designs will work as intended? Ensuring correct functioning of cyber-physical systems is among the most challenging and most important problems in computer science, mathematics, and engineering. But the ability to analyze and understand global system behavior is the key to designing smart and reliable control.

Today, there is a fundamental mismatch between the actual combinations of dynamics that occur in applications and the restricted kinds of dynamics supported in analysis. Safety-critical systems in automotive, aviation, railway, and power grids combine *communication, computation, and control*. Combining computation and control leads to *hybrid systems* [11], whose behavior involves both discrete and continuous dynamics originating, e.g., from discrete control decisions and differential equations of movement. Combining communication and computation leads to *distributed systems* [1], whose dynamics are discrete transitions of system parts that communicate with each other. They may form *dynamic distributed systems*, where the structure of the system is not fixed but evolves over time and agents may appear or disappear during the system evolution.

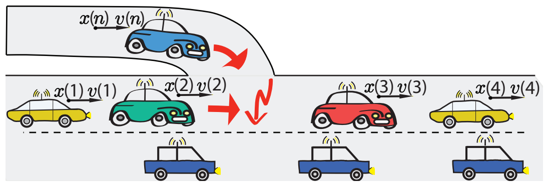

Figure 1: Distributed car control.

Combinations of all three aspects (communication, computation, and control) are used in sophisticated applications, e.g., cooperative distributed car control [13]. Neither structure nor dimension stay the same, because new cars can appear on the street or leave it; see Fig. 1. These systems are *(dynamic) distributed hybrid systems*. They cannot be considered just as a distributed system (because, e.g., the continuous evolution of positions and velocities matters for collision freedom in car control) nor just as a hybrid system (because the evolving system structure and appearance of new agents can make an otherwise collision-free system unsafe). It is generally impossible to split the analysis of distributed hybrid systems soundly into an analysis of a distributed system (without continuous movement) and an analysis of a hybrid system (without structural changes or appearance), because all kinds of dynamics interact. Just like hybrid systems that generally cannot be analyzed from a purely discrete or a purely continuous perspective [11, 17].

Distributed hybrid systems have been considered to varying degrees in modeling languages [6, 21, 15, 16]. In order to build these systems, however, scientists and engineers also need analytic tools to understand and predict their behavior. But formal verification and proof techniques do not yet support the required combination of dynamical effects—which is not surprising given the numerous sources of undecidability for distributed hybrid systems verification.

In this paper, we provide the logical foundations to close this fundamental analytic gap. We develop *quantified hybrid programs* (QHPs) as a model for distributed hybrid systems, which combine dynamical effects from multiple sources: *discrete transitions, continuous evolution, dimension changes, and structural dynamics*. In order to account for changes in the dimension and for co-evolution of an unbounded and evolving number of participants, we generalize the notion of states from assignments for primitive system variables to full first-order structures. Function term $x(i)$ may denote the position of car i of type C, $f(i)$ could be the car registered by communication as the car following car i, and the term $d(i, f(i))$ could denote the minimum safety distance negotiated between car i and its follower. The values of all these terms may evolve *for all* i as time progresses according to interacting laws of discrete and continuous dynamics. They are also affected by changing the system dimension as new cars appear, disappear, or by reconfiguring the system structure dynamically. The defining characteristic of QHPs is that they allow *quantified hybrid dynamics* in which variables like i that occur in function arguments of the system dynamics are quantified over, such that the system co-evolves, e.g., *for all* cars i of type C.

There is a crucial difference between a primitive system variable x and a first-order function term $x(i)$, where i is quantified over. Hybrid dynamics of primitive system variables can model, say, 5 cars (putting scalability issues aside), but not n cars and not systems with a varying number of cars. With first-order function symbols $x(i)$ and hybrid dynamics quantifying over all cars i, a QHP can represent *any* number of cars at once and even (dis)appearance of cars.

Verification of distributed hybrid systems is challenging, because they have three independent sources of undecidability: discrete dynamics, continuous dynamics, and structural/dimensional dynamics. As an analysis tool for distributed hybrid systems, we introduce a specification and verification logic for QHPs that we call *quantified differential dynamic logic* (Qd$\mathcal{L}$). Qd$\mathcal{L}$ provides dynamic logic [10] modal operators $[\alpha]$ and $\langle\alpha\rangle$ that refer to the states reachable by QHP α and can be placed in front of any formula. Formula $[\alpha]\phi$ expresses that all states reachable by system α satisfy formula ϕ, while $\langle\alpha\rangle\phi$ expresses that there is at least one reachable state satisfying ϕ. These modalities can express necessary or possible properties of the transition behavior of α. With its ability to verify (dynamic) distributed hybrid systems and quantified dynamics, Qd$\mathcal{L}$ is a major extension of prior work for static hybrid systems [17, 18] or programs [2, 22].

Our primary contributions are:

- We introduce a *system model and semantics* that succinctly captures the logical quintessence of (dynamic) distributed hybrid systems with joint discrete, continuous, structural, and also dimension-changing dynamics.
- We introduce a *specification/verification logic* for distributed hybrid systems.
- We present a *proof calculus* for this logic, which, to the best of our knowledge, is the *first verification approach* that can handle distributed hybrid systems with their hybrid dynamics and unbounded (and evolving) dimensions.
- We prove that this compositional calculus is a *sound and complete axiomatization* relative to differential equations.
- We have used our proof calculus to verify *collision freedom in a distributed car control system*, where new cars may appear dynamically on the road.

This work constitutes the logical foundation for analysis of distributed hybrid systems. Since distributed hybrid control is the key to control numerous advanced systems, analytic approaches have significant potential for applications.

Our verification approach for distributed hybrid systems is a fundamental extension compared to previous approaches. In much the same way as first-order logic increases the expressive power over propositional logic (quantifiers and function symbols are required to express properties of unbounded structures), $\mathsf{Qd}\mathcal{L}$ increases the expressive power over its predecessors (because first-order functions and quantifiers in the dynamics of QHPs are required to characterize systems with unbounded and changing dimensions).

2 Related Work

Multi-party distributed control has been suggested for car control [13] and air traffic control [7]. Due to limits in verification technology, no formal analysis of the distributed hybrid dynamics has been possible for these systems yet. Analysis results include discrete message handling [13] or collision avoidance for two participants [7]. In distributed car control and air traffic control systems, appearance of new participants is a major unsolved challenge for formal verification.

The importance of understanding dynamic / reconfigurable distributed hybrid systems was recognized in modeling languages SHIFT [6] and R-Charon [15]. They focused on simulation / compilation [6] or the development of a semantics [15], so that no verification is possible yet. For stochastic simulation see [16], where soundness has not been proven, because ensuring coverage is difficult.

For distributed hybrid systems, even giving a formal semantics is very challenging [4, 21, 15, 23]! Zhou et al. [4] gave a semantics for a hybrid version of CSP in the Extended Duration Calculus. Rounds [21] gave a semantics in a rich set theory for a spatial logic for a hybrid version of the π-calculus. In the hybrid π-calculus, processes interact with a continuously changing environment, but cannot themselves evolve continuously, which would be crucial to capture the physical movement of traffic agents. From the semantics alone, no verification is possible in these approaches, except perhaps by manual semantic reasoning.

Other process-algebraic approaches, like χ [23], have been developed for modeling and simulation. Verification is still limited to small fragments that can be translated directly to other verification tools like PHAVer or UPPAAL, which have fixed dimensions and restricted dynamics (no distributed hybrid systems).

Our approach is completely different. It is based on first-order structures and dynamic logic. We focus on developing a logic that supports distributed hybrid dynamics and is amenable to automated theorem proving in the logic itself.

Our previous work and other verification approaches for static hybrid systems cannot verify distributed hybrid systems. Distributed hybrid systems may have an unbounded and changing number of components/participants, which cannot be represented with any fixed number of dimensions of the state space. In distributed car control, for instance, there is no prior limit on the number of cars on the street. Even when there is a limit, explicit replication of the system, say, 100 times, does not yield a scalable verification approach.

Approaches for distributed systems [1] do not cover hybrid systems, because the addition of differential equations to distributed systems is even more challenging than the addition of differential equations to discrete dynamics.

3 Syntax of Qd$\mathcal{L}$

As a formal logic for specifying and verifying correctness properties of distributed hybrid systems, we introduce *quantified differential dynamic logic* (Qd$\mathcal{L}$). Qd$\mathcal{L}$ combines dynamic logic for reasoning about system runs [10] with many-sorted first-order logic for reasoning about all ($\forall i : C\ \phi$) or some ($\exists i : C\ \phi$) objects of a sort C, e.g., the sort of all cars. The most important defining characteristic of Qd$\mathcal{L}$ is that its system model of *quantified hybrid programs* (QHP) supports quantified operations that affect *all* objects of a sort C at once. If C is the sort of cars, QHP $\forall i : C\ a(i) := a(i) + 1$ increases the respective accelerations $a(i)$ of *all cars* i at once. QHP $\forall i : C\ v(i)' = a(i)$ represents a continuous evolution of the respective velocities $v(i)$ of *all cars* at once according to their acceleration. Quantified assignments and quantified differential equation systems are crucial for representing distributed hybrid systems where an unbounded number of objects co-evolve simultaneously. Note that we use the same quantifier notation $\forall i : C$ for quantified operations in programs and for logical formulas.

We model the appearance of new participants in the distributed hybrid system, e.g., new cars entering the road, by a program $n := \mathsf{new}\, C$. It creates a new object of type C, thereby extending the range of subsequent quantified assignments or differential equations ranging over created objects of type C. With quantifiers and function terms, new can be handled in a modular way (Section 5).

3.1 Quantified Differential Dynamic Logic

Sorts Qd$\mathcal{L}$ supports a (finite) number of object sorts, e.g., the sort of all cars. For continuous quantities of distributed hybrid systems like positions or velocities, we add the sort $\mathbb{R}$ for real numbers. See previous work [2] for subtyping of sorts.

Terms Qd$\mathcal{L}$ terms are built from a set of (sorted) function/variable symbols as in many-sorted first-order logic. Unlike in first-order logic, the interpretation of function symbols can change by running QHPs. Even objects may appear or disappear while running QHPs. We use function symbol $\mathsf{E}(\cdot)$ to distinguish between objects i that actually exist and those that have not been created yet, depending on the value of $\mathsf{E}(i)$, which may change its interpretation. We use $0, 1, +, -, \cdot$ with the usual notation and fixed semantics for real arithmetic. For $n \geq 0$ we abbreviate $f(s_1, \ldots, s_n)$ by $f(\vec{s})$ using vectorial notation and we use $\vec{s} = \vec{t}$ for element-wise equality.

Formulas The formulas of Qd$\mathcal{L}$ are defined as in first-order dynamic logic plus many-sorted first-order logic by the following grammar (ϕ, ψ are formulas, θ_1, θ_2 are terms of the same sort, i is a variable of sort C, and α is a QHP):

$$\phi, \psi \ ::= \ \theta_1 = \theta_2 \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \forall i : C\ \phi \mid \exists i : C\ \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

We use standard abbreviations to define $\leq, >, <, \vee, \rightarrow$. Sorts $C \neq \mathbb{R}$ have no ordering and only $\theta_1 = \theta_2$ is allowed. For sort $\mathbb{R}$, we abbreviate $\forall x : \mathbb{R}\ \phi$ by $\forall x\, \phi$. In the following, all formulas and terms have to be well-typed. Qd$\mathcal{L}$ formula $[\alpha]\phi$ expresses that *all states* reachable by QHP α

satisfy formula ϕ. Likewise, $\langle\alpha\rangle\phi$ expresses that *there is at least one state* reachable by α for which ϕ holds.

For short notation, we allow conditional terms of the form $\mathsf{if}\,\phi\,\mathsf{then}\,\theta_1\,\mathsf{else}\,\theta_2\,\mathsf{fi}$ (where θ_1 and θ_2 have the same sort). This term evaluates to θ_1 if the formula ϕ is true and to θ_2 otherwise. We consider formulas with conditional terms as abbreviations, e.g., $\psi(\mathsf{if}\,\phi\,\mathsf{then}\,\theta_1\,\mathsf{else}\,\theta_2\,\mathsf{fi})$ for $(\phi \to \psi(\theta_1)) \land (\neg\phi \to \psi(\theta_2))$.

Example A major challenge in distributed car control systems [13] is that they do not follow fixed, static setups. Instead, new situations can arise dynamically that change structure and dimension of the system whenever new cars appear on the road from on-ramps or leave it; see Fig. 1. As a running example, we model a *distributed car control system DCCS*. First, we consider Qd$\mathcal{L}$ properties.

If i is a term of type C (for cars), let $x(i)$ denote the position of car i, $v(i)$ its current velocity, and $a(i)$ its current acceleration. A state is collision-free if all cars are at different positions, i.e., $\forall i{\neq}j : C\ x(i){\neq}x(j)$. The following Qd$\mathcal{L}$ formula expresses that the system *DCCS* controls cars collision-free:

$$(\forall i,j : C\ \mathcal{M}(i,j)) \ \to\ [\mathit{DCCS}]\ \forall i{\neq}j : C\ x(i){\neq}x(j) \tag{1}$$

It says that *DCCS* controlled cars are always in a collision-free state (postcondition), provided that *DCCS* starts in a state satisfying $\mathcal{M}(i,j)$ for all cars i,j (precondition). Formula $\mathcal{M}(i,j)$ characterizes a simple compatibility condition: for different cars $i \neq j$, the car that is further down the road (i.e., with greater position) neither moves slower nor accelerates slower than the other car, i.e.:

$$\begin{aligned}\mathcal{M}(i,j)\ \equiv\ i \neq j \to \big(&(x(i) < x(j) \land v(i) \le v(j) \land a(i) \le a(j)) \\ &\lor (x(i) > x(j) \land v(i) \ge v(j) \land a(i) \ge a(j))\big)\end{aligned} \tag{2}$$

3.2 Quantified Hybrid Programs

As a system model for distributed hybrid systems, we introduce *quantified hybrid programs* (QHP). These are regular programs from dynamic logic [10] to which we add quantified assignments and quantified differential equation systems for *distributed* hybrid dynamics. From these, QHPs are built like a Kleene algebra with tests [14]. QHPs are defined by the following grammar (α, β are QHPs, θ a term, i a variable of sort C, f is a function symbol, $\vec{s}$ is a vector of terms with sorts compatible to f, and χ is a formula of first-order logic):

$$\alpha, \beta\ ::=\ \forall i : C\ f(\vec{s}) := \theta \mid \forall i : C\ f(\vec{s})' = \theta \,\&\, \chi \mid ?\chi \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^*$$

Quantified State Change The effect of *quantified assignment* $\forall i : C\ f(\vec{s}) := \theta$ is an instantaneous discrete jump assigning θ to $f(\vec{s})$ simultaneously for all objects i of sort C. The effect of *quantified differential equation* $\forall i : C\ f(\vec{s})' = \theta \,\&\, \chi$ is a continuous evolution where, for all objects i of sort C, all differential equations $f(\vec{s})' = \theta$ hold and formula χ holds throughout the evolution (the state remains in the region described by χ). The dynamics of QHPs changes the interpretation of terms over time: $f(\vec{s})'$ is intended to denote the derivative of the interpretation of the term $f(\vec{s})$ over time during continuous evolution, not the derivative of $f(\vec{s})$ by its argument $\vec{s}$. For $f(\vec{s})'$ to be defined, we assume f is an $\mathbb{R}$-valued function symbol. For simplicity, we assume that f does not occur in $\vec{s}$. In most quantified assignments/differential equations $\vec{s}$ is just i. Time itself is implicit. If a clock variable t is needed in a QHP, it can be axiomatized by $t' = 1$, which is equivalent to $\forall i : C\ t' = 1$ where i does not occur in t. For such *vacuous quantification* (i does not occur anywhere), we may omit $\forall i : C$ from assignments and differential equations. Similarly, we may omit vectors $\vec{s}$ of length 0.

Regular Programs The effect of *test* $?\chi$ is a *skip* (i.e., no change) if formula χ is true in the current state and *abort* (blocking the system run by a failed assertion), otherwise. *Nondeterministic choice* $\alpha \cup \beta$ is for alternatives in the behavior of the distributed hybrid system. In the *sequential composition* $\alpha;\beta$, QHP β starts after α finishes (β never starts if α continues indefinitely). *Nondeterministic repetition* α^* repeats α an arbitrary number of times, possibly zero times.

QHPs (with their semantics and our proof rules) can be extended to systems of quantified differential equations, simultaneous assignments to multiple functions f, g, or statements with multiple quantifiers ($\forall i : C\, \forall j : D\ \ldots$). To simplify notation, we do not focus on these cases, which are vectorial extensions [17, 2].

Example Continuous movement of position $x(i)$ of car i with acceleration $a(i)$ is expressed by differential equation $x(i)'' = a(i)$, which corresponds to the first-order differential equation system $x(i)' = v(i), v(i)' = a(i)$ with velocity $v(i)$. Simultaneous movement of all cars with their respective accelerations $a(i)$ is expressed by the QHP $\forall i : C\ (x(i)'' = a(i))$ where quantifier $\forall i : C$ ranges over all cars, such that all cars co-evolve at the same time.

In addition to continuous dynamics, cars have discrete control. In the following QHP, discrete and continuous dynamics interact (repeatedly by the *):

$$\big(\forall i : C\ (a(i) := \mathsf{if}\, \forall j : C\, \mathit{far}(i,j)\ \mathsf{then}\, a\ \mathsf{else}\ -b\, \mathsf{fi});\ \ \forall i : C\ (x(i)'' = a(i))\big)^* \quad (3)$$

First, all cars i control their acceleration $a(i)$. Each car i chooses maximum acceleration $a \geq 0$ for $a(i)$ if its distance to all other cars j is far enough (some condition $\mathit{far}(i,j)$). Otherwise, i chooses full braking $-b < 0$. After all accelerations have been set, all cars move continuously along $\forall i : C\ (x(i)'' = a(i))$. Accelerations may change repeatedly, because the repetition operator * can repeat the QHP when the continuous evolution stops at any time.

4 Semantics of Qd$\mathcal{L}$

The Qd$\mathcal{L}$ semantics is a *constant domain Kripke semantics [9] with first-order structures as states* that associate total functions of appropriate type with function symbols. In constant domain, all states share the same domain for quantifiers. In particular, we choose to represent object creation not by changing the domain of states, but by changing the interpretation of the createdness flag $\mathsf{E}(i)$ of the object denoted by i. With $\mathsf{E}(i)$, object creation is definable (Section 5).

States A *state* σ associates an infinite set $\sigma(C)$ of objects with each sort C, and it associates a function $\sigma(f)$ of appropriate type with each function symbol f, including $\mathsf{E}(\cdot)$. For simplicity, σ also associates a value $\sigma(i)$ of appropriate type with each variable i. The domain of $\mathbb{R}$ and the interpretation of $0, 1, +, -, \cdot$ is that of real arithmetic. We assume constant domain for each sort C: all states σ, τ share the same infinite domains $\sigma(C) = \tau(C)$. Sorts $C \neq D$ are disjoint: $\sigma(C) \cap \sigma(D) = \emptyset$. The set of all states is denoted by $\mathcal{S}$. The state σ_i^e agrees with σ except for the interpretation of variable i, which is changed to e.

Formulas We use $\sigma[\![\theta]\!]$ to denote the value of term θ at state σ. Especially, $\sigma_i^e[\![\theta]\!]$ denotes the value of θ in state σ_i^e, i.e., in state σ with i interpreted as e. Further, $\rho(\alpha)$ denotes the state transition relation of QHP α as defined below. The *interpretation* $\sigma \models \phi$ of Qd$\mathcal{L}$ formula ϕ with respect to state σ is defined as:

1. $\sigma \models (\theta_1 = \theta_2)$ iff $\sigma[\![\theta_1]\!] = \sigma[\![\theta_2]\!]$; accordingly for $\geq$.
2. $\sigma \models \phi \wedge \psi$ iff $\sigma \models \phi$ and $\sigma \models \psi$; accordingly for $\neg$.
3. $\sigma \models \forall i : C\, \phi$ iff $\sigma_i^e \models \phi$ for all objects $e \in \sigma(C)$.
4. $\sigma \models \exists i : C\, \phi$ iff $\sigma_i^e \models \phi$ for some object $e \in \sigma(C)$.
5. $\sigma \models [\alpha]\phi$ iff $\tau \models \phi$ for all states τ with $(\sigma, \tau) \in \rho(\alpha)$.
6. $\sigma \models \langle\alpha\rangle\phi$ iff $\tau \models \phi$ for some τ with $(\sigma, \tau) \in \rho(\alpha)$.

Programs The *transition relation,* $\rho(\alpha) \subseteq \mathcal{S} \times \mathcal{S}$, of QHP α specifies which state $\tau \in \mathcal{S}$ is reachable from $\sigma \in \mathcal{S}$ by running QHP α. It is defined inductively:

1. $(\sigma, \tau) \in \rho(\forall i : C\, f(\vec{s}) := \theta)$ iff state τ is identical to σ except that at each position $\vec{o}$ of f: if $\sigma_i^e[\![\vec{s}]\!] = \vec{o}$ for some object $e \in \sigma(C)$, then $\tau(f)\big(\sigma_i^e[\![\vec{s}]\!]\big) = \sigma_i^e[\![\theta]\!]$. If there are multiple objects e giving the same position $\sigma_i^e[\![\vec{s}]\!] = \vec{o}$, then all of the resulting states τ are reachable.
2. $(\sigma, \tau) \in \rho(\forall i : C\, f(\vec{s})' = \theta \,\&\, \chi)$ iff, there is a function φ:$[0, r] \to \mathcal{S}$ for some $r \geq 0$ with $\varphi(0) = \sigma$ and $\varphi(r) = \tau$ satisfying the following conditions. At each time $t \in [0, r]$, state $\varphi(t)$ is identical to σ, except that at each position $\vec{o}$ of f: if $\sigma_i^e[\![\vec{s}]\!] = \vec{o}$ for some object $e \in \sigma(C)$, then, at each time $\zeta \in [0, r]$:

   - The differential equations hold and derivatives exist (trivial for $r = 0$):
   $$\frac{\mathsf{d}\,(\varphi(t)_i^e[\![f(\vec{s})]\!])}{\mathsf{d}t}(\zeta) = (\varphi(\zeta)_i^e[\![\theta]\!])$$
   - The evolution domain is respected: $\varphi(\zeta)_i^e \models \chi$.

   If there are multiple objects e giving the same position $\sigma_i^e[\![\vec{s}]\!] = \vec{o}$, then all of the resulting states τ are reachable.

3. $\rho(?\chi) = \{(\sigma, \sigma) \ : \ \sigma \models \chi\}$
4. $\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$
5. $\rho(\alpha; \beta) = \{(\sigma, \tau) \ : \ (\sigma, z) \in \rho(\alpha)$ and $(z, \tau) \in \rho(\beta)$ for a state $z\}$
6. $(\sigma, \tau) \in \rho(\alpha^*)$ iff there is an $n \in \mathbb{N}$ with $n \geq 0$ and there are states $\sigma = \sigma_0, \ldots, \sigma_n = \tau$ such that $(\sigma_i, \sigma_{i+1}) \in \rho(\alpha)$ for all $0 \leq i < n$.

The semantics is *explicit change*: nothing changes unless an assignment or differential equation specifies how. In cases 1–2, only f changes and only at positions of the form $\sigma_i^e[\![\vec{s}]\!]$ for some interpretation $e \in \sigma(C)$ of i. If there are multiple such e that affect the same position $\vec{o}$, any of those changes can take effect by a nondeterministic choice. QHP $\forall i : C\ x := a(i)$ may change x to *any* $a(i)$. Hence, $[\forall i : C\ x := a(i)]\phi(x) \equiv \forall i : C\ \phi(a(i))$ and $\langle \forall i : C\ x := a(i) \rangle \phi(x) \equiv \exists i : C\ \phi(a(i))$. Similarly, x can evolve along $\forall i : C\ x' = a(i)$ with any of the slopes $a(i)$. But evolutions cannot start with slope $a(c)$ and then switch to a different slope $a(d)$ later. Any choice for i is possible but i remains unchanged during each evolution.

We call a quantified assignment $\forall i : C\ f(\vec{s}) := \theta$ or a quantified differential equation of the form $\forall i : C\ f(\vec{s})' = \theta \,\&\, \chi$ *injective* iff there is at most one e satisfying cases 1–2. We call quantified assignments and quantified differential equations *schematic* iff $\vec{s}$ is i (thus injective) and the only arguments to function symbols in θ are i. Schematic quantified differential equations like $\forall i : C\ f(i)' = a(i) \,\&\, \chi$ are very common, because distributed hybrid systems often have a family of similar differential equations replicated for multiple participants i. Their synchronization typically comes from discrete communication on top of their continuous dynamics, less often from complicated, physically coupled differential equations.

5 Actual Existence and Object Creation

Actual Existence For the Qd$\mathcal{L}$ semantics, we chose constant domain semantics, i.e., all states share the same domains. Thus quantifiers range over all possible objects (*possibilist quantification*) not just over active existing objects (*actualist quantification* in varying domains) [9]. In order to distinguish between *actual objects* that exist in a state, because they have already been created and can now actively take part in its evolution, versus *possible objects* that still passively await creation, we use function symbol $\mathsf{E}(\cdot)$. Symbol $\mathsf{E}(\cdot)$ is similar to existence predicates in first-order modal logic [9], but its value can be assigned to in QHPs.

Object Creation For term i of type $C \neq \mathbb{R}$, $\mathsf{E}(i) = 1$ represents that the object denoted by i has been created and actually exists. We use $\mathsf{E}(i) = 0$ to represent that i has not been created. Object creation amounts to changing the interpretation of $\mathsf{E}(i)$. For an object denoted by i that has not been created ($\mathsf{E}(i) = 0$), object creation corresponds to the state change caused by assignment $\mathsf{E}(i) := 1$. With quantified assignments and function symbols, *object creation* is definable:

$$n := \mathsf{new}\, C \ \equiv \ (\forall j : C\ n := j);\ ?(\mathsf{E}(n) = 0);\ \mathsf{E}(n) := 1$$

It assigns an arbitrary j of type C to n that did not exist before ($\mathsf{E}(n)=0$) and adjusts existence ($\mathsf{E}(n):=1$). *Disappearance* of object i corresponds to $\mathsf{E}(i):=0$. Our choice avoids semantic subtleties of varying domains about the meaning of free variables denoting non-existent objects as in free logics [9]. Denotation is standard. Terms may just denote objects that have not been activated yet. This is useful to initialize new objects (e.g., $x(n):=8$) before activation ($\mathsf{E}(n):=1$).

Actualist Quantifiers We define abbreviations for *actualist quantifiers* in formulas / quantified assignments / quantified differential equations that range only over previously *created objects*, similar to relativization in modal logic [9]:

$$\begin{aligned}
\forall i:C!\ \phi &\equiv \forall i:C\ (\mathsf{E}(i)=1 \rightarrow \phi)\\
\exists i:C!\ \phi &\equiv \exists i:C\ (\mathsf{E}(i)=1 \land \phi)\\
\forall i:C!\ f(\vec{s}):=\theta &\equiv \forall i:C\ f(\vec{s}):=(\mathsf{if}\,\mathsf{E}(i)=1\,\mathsf{then}\,\theta\,\mathsf{else}\,f(\vec{s})\,\mathsf{fi})\\
\forall i:C!\ f(\vec{s})'=\theta &\equiv \forall i:C\ f(\vec{s})'=(\mathsf{if}\,\mathsf{E}(i)=1\,\mathsf{then}\,\theta\,\mathsf{else}\,0\,\mathsf{fi})\ \equiv\ \forall i:C\ f(\vec{s})'=\mathsf{E}(i)\theta
\end{aligned}$$

The last 2 cases define quantified state change for actually existing objects using conditional terms that choose effect θ if $\mathsf{E}(i)=1$ and choose no effect (retaining the old value $f(\vec{s})$ or evolving with slope 0) if $\mathsf{E}(i)=0$. Notation $C!$ signifies that the quantifier domain is restricted to actually existing objects of type C.

We generally assume that QHPs involve only quantified assignments / differential equations that are restricted to created objects, because real systems only affect objects that are physically present, not those that will be created later. We still treat actualist quantification over $C!$ as a defined notion, in order to simplify the semantics and proof calculus by separating object creation from quantified state change rules in a modular way. If only finitely many objects have been created in the initial state (say 0), then it is easy to see that only finitely many new objects will be created with finitely many such QHP transitions, because each quantified state change for $C!$ only ranges over a finite domain then. We thus assume $\mathsf{E}(\cdot)$ to have *(unbounded but) finite support*, i.e., each state only has a finite number of positions i at which $\mathsf{E}(i)=1$. This makes sense in practice, because there is a varying but still finite numbers of participants (e.g., cars).

Example In order to restrict the dynamics and properties in the car control examples of Section 3 to created and physically present cars, we simply replace each occurrence of $\forall i:C$ with $\forall i:C!$. A challenging feature of distributed car control, however, is that new cars may appear dynamically from on-ramps (Fig. 1) changing the set of active objects. To model this, we consider the following QHP:

$$\mathit{DCCS}\ \equiv\ (n:=\mathsf{new}\,C;\ (?\forall i:C!\ \mathcal{M}(i,n));\ \forall i:C!\ (x(i)''=a(i)))^* \qquad (4)$$

It creates a new car n at an arbitrary position $x(n)$ satisfying compatibility condition $\mathcal{M}(i,n)$ with respect to all other created cars i. Hence *DCCS* allows new cars to appear, but not drop right out of the sky in front of a fast car or run at Mach 8 only 10ft away. When cars appear into the horizon from on-ramps, this condition captures that a car is only allowed to join the lane ("appear" into the model world) if it cannot cause a crash with other existing cars (Fig. 1). Unboundedly many cars may appear during the operation of *DCCS* and change the system dimension arbitrarily, because of the repetition operator *.

DCCS is simple but shows how properties of distributed hybrid systems can be expressed in Qd$\mathcal{L}$. Structural dynamics corresponds to assignments to function terms. Say, $f(i)$ is the car registered by communication as the car following car i. Then a term $d(i, f(i))$, which denotes the minimum safety distance negotiated between car i and its follower, is a crucial part of the system dynamics. Restructuring the system in response to lane change corresponds to assigning a new value to $f(i)$, which impacts the value of $d(i, f(i))$ in the system dynamics.

6 Proof Calculus

In Fig. 2, we present a proof calculus for Qd$\mathcal{L}$ formulas. We use the sequent notation informally for a systematic proof structure. With finite sets of formulas for the *antecedent* Γ and *succedent* Δ, *sequent* $\Gamma \rightarrow \Delta$ is an abbreviation for the formula $\bigwedge_{\phi \in \Gamma} \phi \ \rightarrow \ \bigvee_{\psi \in \Delta} \psi$. The calculus uses standard proof rules for propositional logic with cut rule (not shown). The proof rules are used backwards from the *conclusion* (goal below horizontal bar) to the *premisse*s (subgoals above bar).

In the calculus, we use substitutions that take effect within formulas and programs (defined as usual). Only admissible substitutions are applicable, which is crucial for soundness. An application of a substitution σ is *admissible* if no replaced term θ occurs in the scope of a quantifier or modality binding a symbol in θ or in its replacement $\sigma\theta$. A modality *binds* a symbol f iff it contains an assignment to f (like $\forall i\!:\!C\ f(\vec{s}) := \theta$) or a differential equation containing a $f(\vec{s})'$ (like $\forall i\!:\!C\ f(\vec{s})' = \theta$). The substitutions in Fig. 2 that insert a term θ into $\phi(\theta)$ also have to be admissible for the proof rules to be applicable.

Regular Rules The next proof rules axiomatize sequential composition ($[;]$,$\langle;\rangle$), nondeterministic choice ($[\cup]$,$\langle\cup\rangle$), and test ($[?]$,$\langle?\rangle$) of regular programs as in dynamic logic [10]. Like other rules in Fig. 2, these rules do not contain sequent symbol $\rightarrow$, i.e., they can be applied to any subformula. These rules represent (directed) equivalences: conclusion and premiss are equivalent.

Quantified Differential Equations Rules $[']$,$\langle'\rangle$ handle continuous evolutions for quantified differential equations with first-order definable solutions. Given a solution for the quantified differential equation system with symbolic initial values $f(\vec{s})$, continuous evolution along differential equations can be replaced with a quantified assignment $\forall i\!:\!C\ \mathcal{S}(t)$ corresponding to the solution (footnote 1 in Fig. 2), and an additional quantifier for evolution time t. In $[']$, postcondition ϕ needs to hold *for all* evolution durations t. In $\langle'\rangle$, it needs to hold after *some* duration t. The constraint on χ restricts the continuous evolution to remain in the evolution domain region χ at all intermediate times $\tilde{t} \leq t$.

For schematic cases like $\forall i\!:\!C\ f(i)' = a(i)$, first-order definable solutions can be obtained by adding argument i to first-order definable solutions of the deparametrized version $f' = a$. We only present proof rules for first-order definable solutions of quantified differential equations here. See [18] for other proof rules.

$$([;])\ \frac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi} \qquad ([\cup])\ \frac{[\alpha]\phi\land[\beta]\phi}{[\alpha\cup\beta]\phi} \qquad ([?])\ \frac{\chi\to\psi}{[?\chi]\psi}$$

$$(\langle;\rangle)\ \frac{\langle\alpha\rangle\langle\beta\rangle\phi}{\langle\alpha;\beta\rangle\phi} \qquad (\langle\cup\rangle)\ \frac{\langle\alpha\rangle\phi\lor\langle\beta\rangle\phi}{\langle\alpha\cup\beta\rangle\phi} \qquad (\langle?\rangle)\ \frac{\chi\land\psi}{\langle?\chi\rangle\psi}$$

$$([']) \ \frac{\forall t{\geq}0\,\big((\forall 0{\leq}\tilde t{\leq}t\,[\forall i{:}C\,\mathcal{S}(\tilde t)]\chi)\to[\forall i{:}C\,\mathcal{S}(t)]\phi\big)}{[\forall i{:}C\,f(\vec s)'=\theta\,\&\,\chi]\phi}$$ [1]

$$(\langle'\rangle) \ \frac{\exists t{\geq}0\,\big((\forall 0{\leq}\tilde t{\leq}t\,\langle\forall i{:}C\,\mathcal{S}(\tilde t)\rangle\chi)\land\langle\forall i{:}C\,\mathcal{S}(t)\rangle\phi\big)}{\langle\forall i{:}C\,f(\vec s)'=\theta\,\&\,\chi\rangle\phi}$$ [1]

$$([:=]) \ \frac{\textsf{if }\exists i{:}C\ \vec s=[\mathcal A]\vec u\textsf{ then }\forall i{:}C\ (\vec s=[\mathcal A]\vec u\to\phi(\theta))\textsf{ else }\phi(f([\mathcal A]\vec u))\textsf{ fi}}{\phi([\forall i{:}C\ f(\vec s):=\theta]f(\vec u))}$$ [2]

$$(\langle:=\rangle) \ \frac{\textsf{if }\exists i{:}C\ \vec s=\langle\mathcal A\rangle\vec u\textsf{ then }\exists i{:}C\ (\vec s=\langle\mathcal A\rangle\vec u\land\phi(\theta))\textsf{ else }\phi(f(\langle\mathcal A\rangle\vec u))\textsf{ fi}}{\phi(\langle\forall i{:}C\ f(\vec s):=\theta\rangle f(\vec u))}$$ [2]

$$(skip)\ \frac{\Upsilon([\forall i{:}C\ f(\vec s):=\theta]\vec u)}{[\forall i{:}C\ f(\vec s):=\theta]\,\Upsilon(\vec u)}$$ [3] $$([:*])\ \frac{\forall j{:}C\ \phi(\theta)}{[\forall j{:}C\ n:=\theta]\phi(n)} \qquad (\langle:*\rangle)\ \frac{\exists j{:}C\ \phi(\theta)}{\langle\forall j{:}C\ n:=\theta\rangle\phi(n)}$$

$$(ex)\ \frac{true}{\exists n{:}C\ \mathsf{E}(n)=0}$$

$$(\exists\mathrm{r})\ \frac{\Gamma\to\phi(\theta),\exists x\,\phi(x),\Delta}{\Gamma\to\exists x\,\phi(x),\Delta}$$ [4] $$(\forall\mathrm{r})\ \frac{\Gamma\to\phi(f(X_1,..,X_n)),\Delta}{\Gamma\to\forall x\,\phi(x),\Delta}$$ [5]

$$(\forall\mathrm{l})\ \frac{\Gamma,\phi(\theta),\forall x\,\phi(x)\to\Delta}{\Gamma,\forall x\,\phi(x)\to\Delta}$$ [4] $$(\exists\mathrm{l})\ \frac{\Gamma,\phi(f(X_1,..,X_n))\to\Delta}{\Gamma,\exists x\,\phi(x)\to\Delta}$$ [5]

$$(\mathrm{i}\forall)\ \frac{\mathrm{QE}(\forall X,Y\ (\textsf{if }\vec s=\vec t\textsf{ then }\Phi(X)\to\Psi(X)\textsf{ else }\Phi(X)\to\Psi(Y)\textsf{ fi}))}{\Phi(f(\vec s))\to\Psi(f(\vec t))}$$ [6]

$$(\mathrm{i}\exists)\ \frac{\mathrm{QE}(\exists X\ \bigwedge_i(\Phi_i\to\Psi_i))}{\Phi_1\to\Psi_1\quad\ldots\quad\Phi_n\to\Psi_n}$$ [7]

$$([]gen)\ \frac{\phi\to\psi}{\Gamma,[\alpha]\phi\to[\alpha]\psi,\Delta} \qquad (\langle\rangle gen)\ \frac{\phi\to\psi}{\Gamma,\langle\alpha\rangle\phi\to\langle\alpha\rangle\psi,\Delta} \qquad (ind)\ \frac{\phi\to[\alpha]\phi}{\Gamma,\phi\to[\alpha^*]\phi,\Delta}$$

$$(con)\ \frac{v>0\land\varphi(v)\to\langle\alpha\rangle\varphi(v-1)}{\Gamma,\exists v\,\varphi(v)\to\langle\alpha^*\rangle\exists v{\leq}0\,\varphi(v),\Delta}$$ [8]

[1] $t,\tilde t$ are new variables, $\forall i{:}C\ \mathcal{S}(t)$ is the quantified assignment $\forall i{:}C\ f(\vec s):=y_{\vec s}(t)$ with solutions $y_{\vec s}(t)$ of the (injective) differential equations and $f(\vec s)$ as initial values.

[2] Occurrence $f(\vec u)$ in $\phi(f(\vec u))$ is not in scope of a modality (admissible substitution) and we abbreviate assignment $\forall i{:}C\ f(\vec s):=\theta$ by $\mathcal A$, which is assumed to be injective.

[3] $f\neq\Upsilon$ and the quantified assignment $\forall i{:}C\ f(\vec s):=\theta$ is injective. The same rule applies for $\langle\forall i{:}C\ f(\vec s):=\theta\rangle$ instead of $[\forall i{:}C\ f(\vec s):=\theta]$.

[4] θ is an arbitrary term, often a new logical variable.

[5] f is a new (Skolem) function and $X_1,..,X_n$ are all free logical variables of $\forall x\,\phi(x)$.

[6] X,Y are new variables of sort $\mathbb{R}$. QE needs to be applicable in the premiss.

[7] Among all open branches, the free (existential) logical variable X of sort $\mathbb{R}$ only occurs in the branches $\Phi_i\to\Psi_i$. QE needs to be defined for the formula in the premiss, especially, no Skolem dependencies on X occur.

[8] logical variable v does not occur in α.

Figure 2: Rule schemata of the proof calculus for quantified differential dynamic logic.

Quantified Assignments Rules [:=],$\langle{:=}\rangle$ handle quantified assignments (both are equivalent for the injective case, i.e., a match for at most one i). Their effect depends on whether the quantified assignment $\forall i : C\ f(\vec{s}) := \theta$ *matches* $f(\vec{u})$, i.e., there is a choice for i such that $f(\vec{u})$ is affected by the assignment, because $\vec{u}$ is of the form $\vec{s}$ for some i. If it matches, the premiss uses the term θ assigned to $f(\vec{s})$ instead of $f(\vec{u})$, either for all possible $i : C$ that match $f(\vec{u})$ in case of [:=], or for some of those $i : C$ in case of $\langle{:=}\rangle$. Otherwise, the occurrence of f in $\phi(f(\vec{u}))$ will be left unchanged. Rules [:=],$\langle{:=}\rangle$ make a case distinction on matching by if-then-else. In all cases, the original quantified assignment $\forall i : C\ f(\vec{s}) := \theta$, which we abbreviate by $\mathcal{A}$, will be applied to $\vec{u}$ in the premiss, because the value of argument $\vec{u}$ may also be affected by $\mathcal{A}$, recursively. Rule $skip$ characterizes that quantified assignments to f have no effect on all other operators $\Upsilon \neq f$ (including other function symbols, $\wedge$, if then else fi), so that only argument $\vec{u}$ is affected by prefixing $\mathcal{A}$ but Υ remains unchanged.

Rules [:=],$\langle{:=}\rangle$,$skip$ also apply for assignments without quantifiers, which correspond to vacuous quantification $\forall i : C$ where i does not occur anywhere. Rules [:*],$\langle{:*}\rangle$ reduce nondeterministic assignments to universal or existential quantification. For nondeterministic differential equations, see [18].

Object Creation Given our definition of new C as a QHP from Section 5, object creation can be proven by the other proof rules in Fig. 2. In addition, axiom ex expresses that, for sort $C \neq \mathbb{R}$, there always is a new object n that has not been created yet ($\mathsf{E}(n) = 0$), because domains are infinite.

Quantifiers For quantifiers, we cannot just use standard rules [8], because these are for uninterpreted first-order logic and work by instantiating quantifiers, eagerly as in ground tableaux or lazily by unification as in free variable tableaux [8]. Qd$\mathcal{L}$ is based on first-order logic interpreted over the reals [5]. A formula like $\exists a : \mathbb{R}\ \forall x : \mathbb{R}\ (x^2 + a > 0)$ cannot be proven by instantiating quantifiers but is still valid for reals. Unfortunately, the decision procedure for real arithmetic, *quantifier elimination* (QE) in the theory of real-closed fields [5], cannot be applied to formulas with modalities either, because these are quantified reachability statements. Even in discrete dynamic logic, quantifiers plus modalities make validity Π^1_1-complete [10]. Also QE cannot handle sorts $C \neq \mathbb{R}$.

Instead, our Qd$\mathcal{L}$ proof rules combine quantifier handling of many-sorted logic based on instantiation with theory reasoning by QE for the theory of reals. Figure 2 shows rules for quantifiers that combine with decision procedures for real-closed fields. Classical instantiation is sound for sort $\mathbb{R}$, just incomplete.

Rules $\exists$r and $\forall$l instantiate with arbitrary terms θ, including a new free variable X, where $\exists$r and $\forall$l become the usual γ-rules [8, 9]. Rules $\forall$r and $\exists$l correspond to the δ-rule [8]. As in our previous work [17], rules i$\forall$ and i$\exists$ reintroduce and eliminate quantifiers over $\mathbb{R}$ once QE is applicable, as the remaining constraints are first-order in the respective variables. Unlike in previous work, however, functions and different argument vectors can occur in Qd$\mathcal{L}$. If the argument vectors $\vec{s}$ and $\vec{t}$ in i$\forall$ have the same value, the same variable X can be reintroduced for $f(\vec{s})$ and $f(\vec{t})$, otherwise different variables $X \neq Y$ have to be used. Rule i$\exists$ merges all proof branches containing (existential) variable X, because X has to satisfy all branches simultaneously. It thus has multiple conclusions. See [17] for merging and for lifting QE to the presence of function symbols, including formulas that result from the base theory by substitution.

Global Rules The rules in the last block depend on the truth of their premisses in all states reachable by α, thus the context Γ, Δ cannot be used in the premiss. Rules $[]gen$,$\langle\rangle gen$ are Gödel generalization rules and ind is an induction schema for loops with *inductive invariant* ϕ [10]. Similarly, con generalizes Harel's convergence rule [10] to the hybrid case with decreasing *variant* φ [17].

7 Soundness and Completeness

The verification problem for distributed hybrid systems has *three independent sources* of undecidability. Thus, no verification technique can be effective. Hence, Qd$\mathcal{L}$ cannot be effectively axiomatizable. Both its discrete and its continuous fragments alone are subject to Gödel's incompleteness theorem [17]. The fragment with only structural and dimension-changing dynamics is not effective either, because it can encode two-counter machines. The standard way to show adequacy of proof calculi for problems that are not effective is to prove completeness relative to an oracle for handling a fragment of the logic. Unlike in Cook/Harel relative completeness for discrete programs [10], however, Qd$\mathcal{L}$ cannot be complete relative to the fragment of the data logic ($\mathbb{R}$), because real arithmetic is decidable. Instead, we prove that our Qd$\mathcal{L}$ calculus is a complete axiomatization relative to an oracle for the fragment of Qd$\mathcal{L}$ that has only quantified differential equations in modalities. We replace rules $[']$,$\langle'\rangle$ with an oracle and show that the Qd$\mathcal{L}$ calculus would be complete if only we had complete replacements for $[']$,$\langle'\rangle$. The calculus completely lifts *any* approximation of this oracle to the full Qd$\mathcal{L}$!

Theorem 1 (Axiomatization) *The calculus in Fig. 2 is a sound and complete axiomatization of* Qd$\mathcal{L}$ *relative to quantified differential equations; see [20].*

This shows that properties of distributed hybrid systems can be proven to exactly the same extent to which properties of quantified differential equations can be proven. Proof-theoretically, the Qd$\mathcal{L}$ calculus completely lifts verification techniques for quantified continuous dynamics to distributed hybrid dynamics.

8 Distributed Car Control Verification

With the Qd$\mathcal{L}$ calculus and the compatibility condition $\mathcal{M}(i,j)$ from eqn. (2), we can easily prove collision freedom in the distributed car control system (4):

$$(\forall i,j : C!\, \mathcal{M}(i,j)) \rightarrow \\ [(n := \mathsf{new}\, C; ?\forall i : C!\, \mathcal{M}(i,n); \forall i : C!\, (x(i)'' = a(i)))^*]\, \forall i{\neq}j : C!\, x(i){\neq}x(j) \quad (5)$$

See [20] for a formal Qd$\mathcal{L}$ proof of this Qd$\mathcal{L}$ formula, which proves collision freedom despite dynamic appearance of new cars, following the pattern of (1).

In a similar way, we can prove collision freedom in an advanced distributed car control system that has both dynamic appearance of cars on the road and more flexibility in acceleration and braking choices of the individual cars:

$$\begin{aligned}
&\forall i, j : C!\ \mathcal{M}(i,j) \rightarrow \\
&\quad \big[\big(n := \mathsf{new}\, C;\ ?\forall i : C!\ \mathcal{M}(i,n); \\
&\quad\ \forall i : C!\ a(i) := \mathsf{if}\, \forall j : C\, \mathit{far}(i,j)\, \mathsf{then}\, a\, \mathsf{else}\ -b\, \mathsf{fi}; \\
&\quad\ \tau := 0;\ \forall i : C!\ (x(i)' = v(i), v(i)' = a(i), \tau' = 1 \,\&\, v(i) \geq 0 \land \tau \leq \varepsilon)\big)^* \\
&\quad \big]\, \forall i{\neq}j : C!\ x(i){\neq}x(j)
\end{aligned} \tag{6}$$

The QHP in this Qd$\mathcal{L}$ formula allows all cars to change their respective acceleration freely when all other cars are sufficiently far away; see (3). We choose a condition characterizing that the distributed car control system stays controllable at least ε time units (which is the maximum reaction time of the controller):

$$\mathit{far}(i,j) \ \equiv\ x(j) > x(i) \rightarrow x(j) > x(i) + \frac{v(i)^2 - v(j)^2}{2b} + \Big(\frac{a}{b} + 1\Big)\Big(\frac{a}{2}\varepsilon^2 + \varepsilon v(i)\Big)$$

Similarly, we choose a refined version of compatibility condition $\mathcal{M}(i,j)$ that allows varying velocities $v(i)$ for the cars whenever appropriate safety distances are respected such that the cars can still brake safely at a later point:

$$\begin{aligned}
i \neq j \ \rightarrow \big(&(x(i) < x(j) \land v(i)^2 < v(j)^2 + 2b(x(j) - x(i)) \land v(i) \geq 0 \land v(j) \geq 0) \\
&\lor (x(i) > x(j) \land v(j)^2 < v(i)^2 + 2b(x(i) - x(j)) \land v(i) \geq 0 \land v(j) \geq 0)\big)
\end{aligned}$$

See [20] for a formal Qd$\mathcal{L}$ proof of Qd$\mathcal{L}$ formula (6).

9 Conclusions

We have introduced a system model and semantics for dynamic distributed hybrid systems together with a compositional verification logic and proof calculus. We believe this is the *first formal verification approach for distributed hybrid dynamics*, where structure and dimension of the system can evolve jointly with the discrete and continuous dynamics. We have proven our calculus to be a *sound and complete axiomatization* relative to quantified differential equations. Our calculus proves collision avoidance in distributed car control with dynamic appearance of new cars on the road, which is out of scope for other approaches.

Future work includes modular concurrency in distributed hybrid systems, which is already challenging in discrete programs.

Acknowledgments I want to thank Frank Pfenning for his helpful comments.

References

[1] Paul C. Attie and Nancy A. Lynch. Dynamic input/output automata: A formal model for dynamic systems. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *CONCUR*, volume 2154 of *LNCS*, pages 137–151. Springer, 2001.

[2] Bernhard Beckert and André Platzer. Dynamic logic with non-rigid functions: A basis for object-oriented program verification. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *LNCS*, pages 266–280. Springer, 2006.

[3] V. I. Bogachev. Deterministic and stochastic differential equations in infinite-dimensional spaces. *Acta Appl. Math.*, 40(1):25–93, Jul 1995.

[4] Zhou Chaochen, Wang Ji, and Anders P. Ravn. A formal description of hybrid systems. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems*, volume 1066 of *LNCS*, pages 511–530. Springer, 1995.

[5] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symb. Comput.*, 12(3):299–328, 1991.

[6] Akash Deshpande, Aleks Göllü, and Pravin Varaiya. SHIFT: A formalism and a programming language for dynamic networks of hybrid automata. In *Hybrid Systems*, volume 1273 of *LNCS*, pages 113–133. Springer, 1996.

[7] Gilles Dowek, César Muñoz, and Víctor A. Carreño. Provably safe coordinated strategy for distributed conflict resolution. In *AIAA Proceedings, AIAA-2005-6047*, 2005.

[8] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1996.

[9] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*. Kluwer, 1999.

[10] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, Cambridge, 2000.

[11] Thomas A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292. IEEE, 1996.

[12] João P. Hespanha and Ashish Tiwari, editors. *HSCC*, volume 3927 of *LNCS*. Springer, 2006.

[13] Ann Hsu, Farokh Eskafi, Sonia Sachs, and Pravin Varaiya. Design of platoon maneuver protocols for IVHS. PATH Research Report UCB-ITS-PRR-91-6, UC Berkeley, 1991.

[14] Dexter Kozen. Kleene algebra with tests. *ACM TOPLAS*, 19(3):427–443, 1997.

[15] Fabian Kratz, Oleg Sokolsky, George J. Pappas, and Insup Lee. R-Charon, a modeling language for reconfigurable hybrid systems. In Hespanha and Tiwari [12], pages 392–406.

[16] José Meseguer and Raman Sharykin. Specification and analysis of distributed object-based stochastic hybrid systems. In Hespanha and Tiwari [12], pages 460–475.

[17] André Platzer. Differential dynamic logic for hybrid systems. *J Autom Reas*, 41(2):143–189, 2008.

[18] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput.*, 20(1):309–352, 2010. DOI 10.1093/logcom/exn070.

[19] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010. To appear.

[20] André Platzer. Quantified differential dynamic logic for distributed hybrid systems. Technical Report CMU-CS-10-126, SCS, Carnegie Mellon University, 2010.

[21] William C. Rounds. A spatial logic for the hybrid π-calculus. In Rajeev Alur and George J. Pappas, editors, *HSCC*, volume 2993 of *LNCS*, pages 508–522. Springer, 2004.

[22] Philipp Rümmer. Sequential, parallel, and quantified updates of first-order structures. In Miki Hermann and Andrei Voronkov, editors, *LPAR*, volume 4246 of *LNCS*, pages 422–436. Springer, 2006.

[23] D. A. van Beek, Ka L. Man, Michel A. Reniers, J. E. Rooda, and Ramon R. H. Schiffelers. Syntax and consistent equation semantics of hybrid Chi. *J. Log. Algebr. Program.*, 68(1-2):129–210, 2006.

[24] Wolfgang Walter. *Ordinary Differential Equations*. Springer, 1998.

A Proofs for Distributed Car Control

In this section we prove collision freedom of the simple and the advanced distributed car control system from Section 8. First we introduce derived proof rules to obtain shorter derivations. For short notation, we abbreviate $\mathsf{E}(i) = 1$ by the predicate $\mathsf{E}(i)$, and abbreviate $\mathsf{E}(i) = 0$ by $\neg\mathsf{E}(i)$. Likewise, we write $\mathsf{E}(i) := 1$ as $\mathsf{E}(i) := \top$ where $\top$ is the formula "true".

A.1 Derived Proof Rules

Figure 3 shows derived rules for concise reasoning in common cases. Rule new axiomatizes object creation as a simple consequence of the definition of $\mathsf{new}\,C$ (Section 5). It chooses any n that did not exist before ($\neg\mathsf{E}(n)$) and adjusts existence ($\mathsf{E}(n) := \top$). Object creation gets propagated to actualist quantifiers $\forall i : C!$, $\exists i : C!$ ranging over all created objects, or quantified assignments / differential equations for all created objects by $\nu\forall$,$\nu\exists$,νA.

Derived rules $\nu\forall$,$\nu\exists$,νA characterize the effect of creating objects of type C on actualist quantifiers over type $C!$ (for $\nu\forall$,$\nu\exists$) or on actualist quantified assignments over $C!$ (νA). They commute object creation with quantification, retaining the effect on the new object explicitly. Rule $\nu\forall$ states that the new object denoted by n—which may not have been created before—needs to satisfy $\phi(n)$ too in order for $\forall i : C!\ \phi(i)$ to hold after $\mathsf{E}(n) := \top$ ensures n is created. Dually, $\nu\exists$ states that created object n is an alternative choice for i, in addition to the previous domain of $C!$.

Similarly, rule νA states that, after creating an object of type C, this created object will be affected by actualist quantified assignments ranging over $C!$, so that commuting has to take care of the effect on the new object explicitly. For this common situation where n is adjoined to the range of quantification (n might even have been in the range before, so the union is not always disjoint), we use the following mnemonic abbreviation in the premiss of νA:

$$\begin{aligned} & \forall i : C!\cup\{n\}\ f(\vec{s}) := \theta \\ \equiv\ & \forall i : C\ f(\vec{s}) := \mathsf{if}\ i = n \lor \mathsf{E}(i)\ \mathsf{then}\ \theta\ \mathsf{else}\ f(\vec{s})\ \mathsf{fi} \end{aligned}$$

$$(\nu\forall)\ \frac{[\![\mathsf{E}(n) := \top]\!]\phi(n) \land \forall i : C!\ [\![\mathsf{E}(n) := \top]\!]\phi(i)}{[\![\mathsf{E}(n) := \top]\!]\forall i : C!\ \phi(i)}\,1 \qquad (\nu A)\ \frac{[\![\forall i : C!\cup\{n\}\ f(\vec{s}) := \theta]\!][\![\mathsf{E}(n) := \top]\!]\phi}{[\![\mathsf{E}(n) := \top]\!][\![\forall i : C!\ f(\vec{s}) := \theta]\!]\phi}\,1$$

$$(\nu\exists)\ \frac{[\![\mathsf{E}(n) := \top]\!]\phi(n) \lor \exists i : C!\ [\![\mathsf{E}(n) := \top]\!]\phi(i)}{[\![\mathsf{E}(n) := \top]\!]\exists i : C!\ \phi(i)}\,1 \qquad (new)\ \frac{\forall n : C\ (\neg\mathsf{E}(n) \to [\![\mathsf{E}(n) := \top]\!]\phi)}{[\![n := \mathsf{new}\,C]\!]\phi}$$

[1] n is of type C

Figure 3: Derived proof rules for object creation in quantified differential dynamic logic.

Proposition 1 *The proof rules in Fig. 3 are derived rules.*

Proof: In order to prove Proposition 1, we show that the respective proof rules in Fig. 3 can be derived from the proof rules in Fig. 2 with standard propositional and first-order reasoning. Rule $\nu\forall$ is a derived rule:

$$\begin{array}{rl}
 & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(n) \land \forall i : C!\ \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i)\\ \hline
 & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(n) \land \forall i : C\ (\mathsf{E}(i) \to \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i))\\ \hline
 & \forall i : C\ \big((i = n \to \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i)) \land (i \neq n \to (\mathsf{E}(i) \to \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i)))\big)\\ \hline
 & \forall i : C\ (\mathsf{if}\ i = n\ \mathsf{then}\ \top\ \mathsf{else}\ \mathsf{E}(i)\ \mathsf{fi} \to \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i))\\ \hline
{\scriptstyle [:=],\langle:=\rangle} & \forall i : C\ (\langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\mathsf{E}(i) \to \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i))\\ \hline
{\scriptstyle skip,skip} & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\forall i : C\ (\mathsf{E}(i) \to \phi(i))\\ \hline
 & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\forall i : C!\ \phi(i)
\end{array}$$

This proof uses simple boolean reasoning (formally: cut) to substitute in $i = n$ in the left conjunct and remove $i \neq n$ on the right conjunct ($i = n$ can be covered in both conjuncts).

Rule $\nu\exists$ is a derived rule with a similar proof:

$$\begin{array}{rl}
 & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(n) \lor \exists i : C!\ \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i)\\ \hline
 & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(n) \lor \exists i : C\ (\mathsf{E}(i) \land \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i))\\ \hline
 & \exists i : C\ \big((i = n \land \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i)) \lor (i \neq n \land \mathsf{E}(i) \land \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i))\big)\\ \hline
 & \exists i : C\ \big((i = n \to \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i)) \land (i \neq n \to \mathsf{E}(i) \land \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i))\big)\\ \hline
 & \exists i : C\ (\mathsf{if}\ i = n\ \mathsf{then}\ \top\ \mathsf{else}\ \mathsf{E}(i)\ \mathsf{fi} \land \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i))\\ \hline
{\scriptstyle [:=],\langle:=\rangle} & \exists i : C\ (\langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\mathsf{E}(i) \land \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi(i))\\ \hline
{\scriptstyle skip,skip} & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\exists i : C\ (\mathsf{E}(i) \land \phi(i))\\ \hline
 & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\exists i : C!\ \phi(i)
\end{array}$$

Assuming $\mathsf{E}()$ does not occur in the assignment $f(\vec{s}) := \theta$, the rule νA is a derived proof rule:

$$\begin{array}{rl}
 & \langle\!\langle \forall i : C!\cup\{n\}\ f(\vec{s}) := \theta\rangle\!\rangle\langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi\\ \hline
 & \langle\!\langle \forall i : C\ f(\vec{s}) := \mathsf{if}\ i = n \lor \mathsf{E}(i)\ \mathsf{then}\ \theta\ \mathsf{else}\ f(\vec{s})\ \mathsf{fi}\rangle\!\rangle\langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi\\ \hline
 & \langle\!\langle \forall i : C\ f(\vec{s}) := \mathsf{if}\ (\mathsf{if}\ i = n\ \mathsf{then}\ \top\ \mathsf{else}\ \mathsf{E}(i)\ \mathsf{fi})\ \mathsf{then}\ \theta\ \mathsf{else}\ f(\vec{s})\ \mathsf{fi}\rangle\!\rangle\langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\phi\\ \hline
{\scriptstyle [:=],\langle:=\rangle} & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\langle\!\langle \forall i : C\ f(\vec{s}) := \mathsf{if}\ \mathsf{E}(i)\ \mathsf{then}\ \theta\ \mathsf{else}\ f(\vec{s})\ \mathsf{fi}\rangle\!\rangle\phi\\ \hline
 & \langle\!\langle \mathsf{E}(n) := \top\rangle\!\rangle\langle\!\langle \forall i : C!\ f(\vec{s}) := \theta\rangle\!\rangle\phi
\end{array}$$

If, instead, $\mathsf{E}()$ does occur in $f(\vec{s}) := \theta$, then there is a similar derived rule where $\vec{s}$ and θ are affected accordingly. Rule new is a derived rule by the definition of $\mathsf{new}\,C$ as a QHP from Section 5, using ex to show existence of an n for $\langle n := \mathsf{new}\,C\rangle\phi$. $\square$

A.2 Proofs for Simple Distributed Car Control

In the Qd$\mathcal{L}$ calculus, we prove collision freedom for the simpler distributed car control system, i.e., Qd$\mathcal{L}$ formula (5) from Section 8, even in the presence of dynamic appearance of new cars on the road. The proof is shown in Fig. 4. It uses induction rule ind with invariant $\forall i, j : C!\ \mathcal{M}(i, j)$. We recall the following abbreviations:

$$\mathcal{M}(i, j) \equiv\ i \neq j \to \big((x(i) < x(j) \land v(i) \leq v(j) \land a(i) \leq a(j)) \lor (x(i) > x(j) \land v(i) \geq v(j) \land a(i) \geq$$
$$\forall i : C!\ (x(i)'' = a(i)) \equiv\ \forall i : C!\ (x(i)' = v(i), v(i)' = a(i))$$

We use the notation $[\![\alpha]\!]\phi$ as synonymous for $[\alpha]\phi$ here just to have more readable bracket grouping. Figure 4 does not show the branch proving that the invariant $\forall i, j : C!\ \mathcal{M}(i,j)$ implies the postcondition $\forall i{\neq}j : C!\ x(i){\neq}x(j)$, which is simple to show.

$$
\begin{array}{ll|ll}
 & * \text{ by QE} & & * \text{ by QE} \\ \hline
\text{i}\forall & , \mathcal{M}_{i,n}, t{\geq}0 \to \mathcal{S}_t\mathcal{M}_{i,n} & \text{i}\forall & \mathcal{M}_{i,j}, \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to \mathcal{S}_t\mathcal{M}_{i,j} \\ \hline
\text{i}\forall & , \mathcal{M}(i,n), t{\geq}0 \to \mathcal{S}_t\mathcal{M}(i,n) & \text{i}\forall & \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to \mathcal{S}_t\mathcal{M}(i,j) \\ \hline
\forall\text{l} & \ldots, \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to \mathcal{S}_t\mathcal{M}(i,n) & \forall\text{l} & \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to \mathcal{S}_t\mathcal{M}(i,j)
\end{array}
$$

$$
\begin{array}{ll}
\hline
\wedge\text{r} & \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to \mathcal{S}_t\mathcal{M}(i,n) \wedge \mathcal{S}_t\mathcal{M}(i,j) \\ \hline
[:=] & \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to [\![\forall i : C!{\cup}\{n\}\ \mathcal{S}_t(i)]\!](\mathcal{M}(i,n) \wedge \mathcal{M}(i,j)) \\ \hline
\forall\text{r} & \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to [\![\forall i : C!{\cup}\{n\}\ \mathcal{S}_t(i)]\!]\forall i,j : C!\ (\mathcal{M}(i,n) \wedge \mathcal{M}(i,j)) \\ \hline
 & \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to [\![\forall i : C!{\cup}\{n\}\ \mathcal{S}_t(i)]\!](\forall i : C!\ \mathcal{M}(i,n) \wedge \mathcal{M}(n,n) \wedge \forall i,j : C!\ \mathcal{M}(i,j) \wedge \forall j : C!\ \mathcal{M}(n,j)) \\ \hline
\nu\forall & \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to [\![\forall i : C!{\cup}\{n\}\ \mathcal{S}_t(i)]\!][\mathsf{E}(n) := 1]\forall i,j : C!\ \mathcal{M}(i,j) \\ \hline
\nu A & \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t{\geq}0 \to [\mathsf{E}(n) := 1][\![\forall i : C!\ \mathcal{S}_t(i)]\!]\forall i,j : C!\ \mathcal{M}(i,j) \\ \hline
\forall\text{r},\to\text{r} & \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n) \to [\mathsf{E}(n) := 1]\forall t{\geq}0\ [\![\forall i : C!\ \mathcal{S}_t(i)]\!]\forall i,j : C!\ \mathcal{M}(i,j) \\ \hline
[{}'] & \forall i,j : C!\ \mathcal{M}(i,j), \mathcal{M}(n,n), \forall i : C!\ \mathcal{M}(i,n) \to [\mathsf{E}(n) := 1][\![\forall i : C!\ (x(i)'' = a(i))]\!]\forall i,j : C!\ \mathcal{M}(i,j) \\ \hline
\nu\forall,\wedge\text{l} & \forall i,j : C!\ \mathcal{M}(i,j), [\mathsf{E}(n) := 1]\forall i : C!\ \mathcal{M}(i,n) \to [\mathsf{E}(n) := 1][\![\forall i : C!\ (x(i)'' = a(i))]\!]\forall i,j : C!\ \mathcal{M}(i,j) \\ \hline
\to\text{r} & \forall i,j : C!\ \mathcal{M}(i,j) \to [\mathsf{E}(n) := 1](\forall i : C!\ \mathcal{M}(i,n) \to [\![\forall i : C!\ (x(i)'' = a(i))]\!]\forall i,j : C!\ \mathcal{M}(i,j)) \\ \hline
[?] & \mathsf{E}(n) = 0, \forall i,j : C!\ \mathcal{M}(i,j) \to [\mathsf{E}(n) := 1][\![?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ (x(i)'' = a(i))]\!]\forall i,j : C!\ \mathcal{M}(i,j) \\ \hline
new & \forall i,j : C!\ \mathcal{M}(i,j) \to [n := \mathsf{new}\,C][\![?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ (x(i)'' = a(i))]\!]\forall i,j : C!\ \mathcal{M}(i,j) \\ \hline
[;] & \forall i,j : C!\ \mathcal{M}(i,j) \to [\![n := \mathsf{new}\,C; ?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ (x(i)'' = a(i))]\!]\forall i,j : C!\ \mathcal{M}(i,j) \\ \hline
ind & \forall i,j : C!\ \mathcal{M}(i,j) \to [\![(n := \mathsf{new}\,C; ?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ (x(i)'' = a(i)))^*]\!]\forall i{\neq}j : C!\ x(i){\neq}x(j)
\end{array}
$$

$$
\begin{aligned}
\mathcal{M}(i,j) &\equiv\ i \neq j \to \big((x(i) < x(j) \wedge v(i) \leq v(j) \wedge a(i) \leq a(j)) \vee (x(i) > x(j) \wedge v(i) \geq v(j) \wedge a(i) \geq a(j))\big) \\
\mathcal{M}_{i,j} &\equiv\ \mathsf{if}\, i = j\, \mathsf{then}\, i \neq i \to \big((\textsc{x}_i < \textsc{x}_i \wedge \textsc{v}_i \leq \textsc{v}_i \wedge \textsc{a}_i \leq \textsc{a}_i) \vee (\textsc{x}_i > \textsc{x}_i \wedge \textsc{v}_i \geq \textsc{v}_i \wedge \textsc{a}_i \geq \textsc{a}_i)\big) \\
&\qquad \mathsf{else}\, i \neq j \to \big((\textsc{x}_i < \textsc{x}_j \wedge \textsc{v}_i \leq \textsc{v}_j \wedge \textsc{a}_i \leq \textsc{a}_j) \vee (\textsc{x}_i > \textsc{x}_j \wedge \textsc{v}_i \geq \textsc{v}_j \wedge \textsc{a}_i \geq \textsc{a}_j)\big)\ \mathsf{fi} \\
&\equiv\ i \neq j \to \big((\textsc{x}_i < \textsc{x}_j \wedge \textsc{v}_i \leq \textsc{v}_j \wedge \textsc{a}_i \leq \textsc{a}_j) \vee (\textsc{x}_i > \textsc{x}_j \wedge \textsc{v}_i \geq \textsc{v}_j \wedge \textsc{a}_i \geq \textsc{a}_j)\big) \\
\mathcal{S}_t(i) &\equiv\ x(i) := x(i) + v(i)t + \frac{a(i)}{2}t^2 \wedge v(i) := v(i) + a(i)t \\
\mathcal{S}_t\mathcal{M}(i,j) &\equiv\ i \neq j\ \to\ \Big((x(i) + v(i)t + \frac{a(i)}{2}t^2 < x(j) + v(j)t + \frac{a(j)}{2}t^2 \wedge v(i) + a(i)t \leq v(j) + a(j)t \wedge a(i) \leq a(j)) \\
&\qquad \vee (x(i) + v(i)t + \frac{a(i)}{2}t^2 > x(j) + v(j)t + \frac{a(j)}{2}t^2 \wedge v(i) + a(i)t \geq v(j) + a(j)t \wedge a(i) \geq a(j))\Big) \\
\mathcal{S}_t\mathcal{M}_{i,j} &\equiv i \neq j\ \to\ \Big((\textsc{x}_i + \textsc{v}_i t + \frac{\textsc{a}_i}{2}t^2 < \textsc{x}_j + \textsc{v}_j t + \frac{\textsc{a}_j}{2}t^2 \wedge \textsc{v}_i + \textsc{a}_i t \leq \textsc{v}_j + \textsc{a}_j t \wedge \textsc{a}_i \leq \textsc{a}_j) \\
&\qquad \vee (\textsc{x}_i + \textsc{v}_i t + \frac{\textsc{a}_i}{2}t^2 > \textsc{x}_j + \textsc{v}_j t + \frac{\textsc{a}_j}{2}t^2 \wedge \textsc{v}_i + \textsc{a}_i t \geq \textsc{v}_j + \textsc{a}_j t \wedge \textsc{a}_i \geq \textsc{a}_j)\Big)
\end{aligned}
$$

Figure 4: Qd$\mathcal{L}$ proof for collision freedom in distributed car control.

DCCS Verification To save space in the car proof, we directly apply proof rules $\forall$r,$\to$r after the *new* rule in Fig. 4. Furthermore, the top-most $\nu\forall$ rule application in Fig. 4 and its subsequent simplification step

$$
\nu\forall\ \frac{(\mathcal{M}(n,n) \wedge \forall i : C!\ \mathcal{M}(i,n) \wedge \forall j : C!\ \mathcal{M}(n,j) \wedge \forall i,j : C!\ \mathcal{M}(i,j))}{[\mathsf{E}(n) := 1]\forall i,j : C!\ \mathcal{M}(i,j)}
$$

is justified by

$$\begin{array}{rc}
& (\mathcal{M}(n,n) \land \forall i\!:\!C!\ \mathcal{M}(i,n) \land \forall j\!:\!C!\ \mathcal{M}(n,j) \land \forall i,j\!:\!C!\ \mathcal{M}(i,j)) \\ \cline{2-2}
& \mathcal{M}(n,n) \land \forall j\!:\!C!\ \mathcal{M}(n,j)\ \land\ \forall i\!:\!C!\ (\mathcal{M}(i,n) \land \forall j\!:\!C!\ \mathcal{M}(i,j)) \\ \cline{2-2}
{\scriptstyle \nu\forall} & [\mathsf{E}(n) := 1]\forall j\!:\!C!\ \mathcal{M}(n,j)\ \land\ \forall i\!:\!C!\ [\mathsf{E}(n) := 1]\forall j\!:\!C!\ \mathcal{M}(i,j) \\ \cline{2-2}
{\scriptstyle \nu\forall} & [\mathsf{E}(n) := 1]\forall i\!:\!C!\ \forall j\!:\!C!\ \mathcal{M}(i,j)
\end{array}$$

There, $\mathcal{M}(n,n)$ simplifies to $true$, because it assumes $n \neq n$. Likewise, $\forall j\!:\!C!\ \mathcal{M}(n,j)$ is subsumed by $\forall i\!:\!C!\ \mathcal{M}(i,n)$, because $\mathcal{M}(i,j)$ is equivalent to $\mathcal{M}(j,i)$.

A.3 Proofs for Advanced Distributed Car Control with Acceleration

In this section, we show a formal Qd$\mathcal{L}$ proof of the Qd$\mathcal{L}$ formula (6) for the advanced distributed car control system. See Fig. 5 on page 21 for a proof of distributed car control in the presence of free acceleration and braking, regulated only by the safety distances to other cars. The proof uses induction rule ind with invariant $\forall i,j\!:\!C!\ \mathcal{M}(i,j)$. We use the notation $[\![\alpha]\!]\phi$ as synonymous for $[\alpha]\phi$ here just to have more readable bracket grouping.

The major difference of the distributed car dynamics considered on page 21 compared to Fig. 4 is that the operation $\forall i\!:\!C!\ \mathcal{A}(i)$ adjusts the respective accelerations $a(i)$ of all cars i differently depending on the distance to other cars. In particular, the cars do no longer have monotonically increasing velocities and accelerations. For each car i, however, maximum acceleration $a(i) := a$ is only permitted when the safety distance is big enough, otherwise the car brakes by $a(i) := -b$ (for maximum acceleration constant $a \geq 0$ and maximum braking force $b > 0$).

The safety condition in $\mathcal{A}(i)$ for an acceleration choice of car i is that for cars j further down the road (greater positions $x(j) > x(i)$), the distance $x(j) - x(i)$ must be large enough so that car i can safely accelerate for up to ε time units and still keep the safety distance to j in the future by appropriate braking, in spite of the increased velocity. This safety distance depends on the respective velocities $v(i)$ and $v(j)$ as well as a, b, ε. Parameter ε is the maximum reaction time for reacting to situation changes, which results from sensor polling frequencies, worst-case computation times, and latencies in actuator activation. Every control cycle is restricted to take at most ε time units by the evolution domain restriction $\tau \leq \varepsilon$ in the definition of the continuous dynamics that we abbreviate by $\forall i\!:\!C!\ (x(i)'' = a(i))$ as defined on page 21. The definition of the compatibility condition $\mathcal{M}(i,j)$ is adapted correspondingly to take into account that the cars may move with completely different accelerations, if only the safety distances are compatible with the different velocities:

$$\begin{aligned}
\mathcal{M}(i,j) \equiv\ i \neq j \to \big(&(x(i) < x(j) \land v(i)^2 < v(j)^2 + 2b(x(j) - x(i)) \land v(i) \geq 0 \land v(j) \geq 0) \\
&\lor (x(i) > x(j) \land v(j)^2 < v(i)^2 + 2b(x(i) - x(j)) \land v(i) \geq 0 \land v(j) \geq 0)\big)
\end{aligned}$$

In fact, any acceleration between $-b$ and a could also be chosen safely in the acceleration case of $\mathcal{A}(i)$, which only makes the proof slightly more complicated.

A.4 Existence and Uniqueness

Existence/uniqueness of solutions by Picard-Lindelöf / Cauchy-Lipschitz theorem [24, Theorem 10.VI] and by Peano theorem [24, Theorem 10.IX] carry over to case 2 of the the semantics $\rho(\alpha)$ in Section 4

$$\dfrac{\dfrac{\dfrac{* \text{ by QE}}{, \mathcal{M}_{i,n}, t\geq 0 \to \mathcal{AS}_t\mathcal{M}_{i,n}}\,\text{i}\forall}{, \mathcal{M}(i,n), t\geq 0 \to \mathcal{AS}_t\mathcal{M}(i,n)}\,\text{i}\forall}{, \forall i : C!\ \mathcal{M}(i,n), t\geq 0 \to \mathcal{AS}_t\mathcal{M}(i,n)}\,\forall\text{l} \qquad \dfrac{\dfrac{\dfrac{* \text{ by QE}}{\mathcal{M}_{i,j}, , t\geq 0 \to \mathcal{AS}_t\mathcal{M}_{i,j}}\,\text{i}\forall}{\mathcal{M}(i,j), , t\geq 0 \to \mathcal{AS}_t\mathcal{M}(i,j)}\,\text{i}\forall}{\forall i,j : C!\ \mathcal{M}(i,j), , t\geq 0 \to \mathcal{AS}_t\mathcal{M}(i,j)}\,\forall\text{l}$$

$$\wedge\text{r}\quad \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t\geq 0 \to \mathcal{AS}_t\mathcal{M}(i,n) \wedge \mathcal{AS}_t\mathcal{M}(i,j)$$

$$[:=]\quad \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t\geq 0 \to \llbracket \forall i : C!\cup\{n\}\ \mathcal{A}(i)\rrbracket \llbracket \forall i : C!\cup\{n\}\ \mathcal{S}_t(i)\rrbracket (\mathcal{M}(i,n) \wedge \mathcal{M}(i,j))$$

$$\forall\text{r}\quad \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t\geq 0 \to \llbracket \forall i : C!\cup\{n\}\ \mathcal{A}(i)\rrbracket \llbracket \forall i : C!\cup\{n\}\ \mathcal{S}_t(i)\rrbracket \forall i,j : C!\ (\mathcal{M}(i,n) \wedge \mathcal{M}(i,j))$$

$$\forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t\geq 0 \to \llbracket \forall i : C!\cup\{n\}\ \mathcal{A}(i)\rrbracket \llbracket \forall i : C!\cup\{n\}\ \mathcal{S}_t(i)\rrbracket (\forall i : C!\ \mathcal{M}(i,n) \wedge \forall i,j : C!\ \mathcal{M}(i,j))$$

$$\nu\forall\quad \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t\geq 0 \to \llbracket \forall i : C!\cup\{n\}\ \mathcal{A}(i)\rrbracket \llbracket \forall i : C!\cup\{n\}\ \mathcal{S}_t(i)\rrbracket [\mathsf{E}(n) := 1] \forall i,j : C!\ \mathcal{M}(i,j)$$

$$\nu A\quad \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n), t\geq 0 \to \llbracket \forall i : C!\cup\{n\}\ \mathcal{A}(i)\rrbracket [\mathsf{E}(n) := 1] \llbracket \forall i : C!\ \mathcal{S}_t(i)\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$\forall\text{r},\to\text{r}\quad \forall i,j : C!\ \mathcal{M}(i,j), \forall i : C!\ \mathcal{M}(i,n) \to \llbracket \forall i : C!\cup\{n\}\ \mathcal{A}(i)\rrbracket [\mathsf{E}(n) := 1] \forall t\geq 0\ \llbracket \forall i : C!\ \mathcal{S}_t(i)\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$[']\quad \forall i,j : C!\ \mathcal{M}(i,j), , \forall i : C!\ \mathcal{M}(i,n) \to \llbracket \forall i : C!\cup\{n\}\ \mathcal{A}(i)\rrbracket [\mathsf{E}(n) := 1] \llbracket \forall i : C!\ (x(i)'' = a(i))\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$\nu A\quad \forall i,j : C!\ \mathcal{M}(i,j), , \forall i : C!\ \mathcal{M}(i,n) \to [\mathsf{E}(n) := 1] \langle \forall i : C!\ \mathcal{A}(i)\rangle \llbracket \forall i : C!\ (x(i)'' = a(i))\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$[;]\quad \forall i,j : C!\ \mathcal{M}(i,j), , \forall i : C!\ \mathcal{M}(i,n) \to [\mathsf{E}(n) := 1] \llbracket \forall i : C!\ \mathcal{A}(i); \forall i : C!\ (x(i)'' = a(i))\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$\nu\forall,\wedge\text{l}\quad \forall i,j : C!\ \mathcal{M}(i,j), [\mathsf{E}(n) := 1] \forall i : C!\ \mathcal{M}(i,n) \to [\mathsf{E}(n) := 1] \llbracket \forall i : C!\ \mathcal{A}(i); \forall i : C!\ (x(i)'' = a(i))\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$\to\text{r}\quad \forall i,j : C!\ \mathcal{M}(i,j) \to [\mathsf{E}(n) := 1] (\forall i : C!\ \mathcal{M}(i,n) \to \llbracket \forall i : C!\ \mathcal{A}(i); \forall i : C!\ (x(i)'' = a(i))\rrbracket \forall i,j : C!\ \mathcal{M}(i,j))$$

$$[?]\quad \mathsf{E}(n) = 0, \forall i,j : C!\ \mathcal{M}(i,j) \to [\mathsf{E}(n) := 1] \llbracket ?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ \mathcal{A}(i); \forall i : C!\ (x(i)'' = a(i))\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$new\quad \forall i,j : C!\ \mathcal{M}(i,j) \to [n := \mathsf{new}\, C] \llbracket ?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ \mathcal{A}(i); \forall i : C!\ (x(i)'' = a(i))\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$[;]\quad \forall i,j : C!\ \mathcal{M}(i,j) \to \llbracket n := \mathsf{new}\, C; ?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ \mathcal{A}(i); \forall i : C!\ (x(i)'' = a(i))\rrbracket \forall i,j : C!\ \mathcal{M}(i,j)$$

$$ind\quad \forall i,j : C!\ \mathcal{M}(i,j) \to \llbracket (n := \mathsf{new}\, C; ?\forall i : C!\ \mathcal{M}(i,n); \forall i : C!\ \mathcal{A}(i); \forall i : C!\ (x(i)'' = a(i)))^* \rrbracket \forall i\neq j : C!\ x(i)\neq x(j)$$

$$\forall i : C!\ (x(i)'' = a(i)) \;\equiv\; \tau := 0;\ \forall i : C!\ (x(i)' = v(i), v(i)' = a(i), \tau' = 1\ \&\ v(i) \geq 0 \wedge \tau \leq \varepsilon)$$

$$\mathcal{A}(i) \equiv\; a(i) := \text{if}\, \forall j : C\ \left(x(j) > x(i) \to x(j) > x(i) + \frac{v(i)^2 - v(j)^2}{2b} + \left(\frac{a}{b} + 1\right)\left(\frac{a}{2}\varepsilon^2 + \varepsilon v(i)\right)\right) \text{ then } a \text{ else } -b \text{ fi}$$

$$\begin{aligned}\mathcal{M}(i,j) \equiv\;& i \neq j \to \big((x(i) < x(j) \wedge v(i)^2 < v(j)^2 + 2b(x(j) - x(i)) \wedge v(i), v(j) \geq 0)\\ &\quad \vee (x(i) > x(j) \wedge v(j)^2 < v(i)^2 + 2b(x(i) - x(j)) \wedge v(i), v(j) \geq 0)\big)\end{aligned}$$

$$\begin{aligned}\mathcal{M}_{i,j} \equiv\;& \text{if}\, i = j \text{ then } i \neq i \to \big((\mathrm{x}_i < \mathrm{x}_i \wedge \mathrm{v}_i^2 < \mathrm{v}_i^2 + 2b(\mathrm{x}_i - \mathrm{x}_i) \wedge \mathrm{v}_i, \mathrm{v}_i \geq 0) \vee (\mathrm{x}_i > \mathrm{x}_i \wedge \mathrm{v}_i^2 < \mathrm{v}_i^2 + 2b(\mathrm{x}_i - \mathrm{x}_i) \wedge \mathrm{v}_i, \mathrm{v}_i \geq 0)\big)\\ & \text{else } i \neq j \to \big((\mathrm{x}_i < \mathrm{x}_j \wedge \mathrm{v}_i^2 < \mathrm{v}_j^2 + 2b(\mathrm{x}_j - \mathrm{x}_i) \wedge \mathrm{v}_i, \mathrm{v}_j \geq 0) \vee (\mathrm{x}_i > \mathrm{x}_j \wedge \mathrm{v}_j^2 < \mathrm{v}_i^2 + 2b(\mathrm{x}_i - \mathrm{x}_j) \wedge \mathrm{v}_i, \mathrm{v}_j \geq 0)\big) \text{ fi}\\ \equiv\;& i \neq j \to \big((\mathrm{x}_i < \mathrm{x}_j \wedge \mathrm{v}_i^2 < \mathrm{v}_j^2 + 2b(\mathrm{x}_j - \mathrm{x}_i) \wedge \mathrm{v}_i, \mathrm{v}_j \geq 0) \vee (\mathrm{x}_i > \mathrm{x}_j \wedge \mathrm{v}_j^2 < \mathrm{v}_i^2 + 2b(\mathrm{x}_i - \mathrm{x}_j) \wedge \mathrm{v}_i, \mathrm{v}_j \geq 0)\big)\end{aligned}$$

$$\mathcal{S}_t(i) \equiv\; x(i) := x(i) + v(i)t + \frac{a(i)}{2}t^2 \wedge v(i) := v(i) + a(i)t$$

$\mathcal{AS}_t\mathcal{M}(i,j) \equiv$ is defined as follows

$$i \neq j \to (\text{if}\, \forall j : C!\ \left(\left(x(j) > x(i) \to x(j) > x(i) + \frac{v(i)^2 - v(j)^2}{2b} + \left(\frac{a}{b} + 1\right)\left(\frac{a}{2}\varepsilon^2 + \varepsilon v(i)\right)\right)\right) \text{ then}$$

$$\text{if}\, \forall i : C!\ \left(\left(x(i) > x(j) \to x(i) > x(j) + \frac{v(j)^2 - v(i)^2}{2b} + \left(\frac{a}{b} + 1\right)\left(\frac{a}{2}\varepsilon^2 + \varepsilon v(j)\right)\right)\right) \text{ then}$$

$$(x(i) + v(i)t + \frac{a}{2}t^2 < x(j) + v(j)t + \frac{a}{2}t^2 \wedge v(i) + atv(i)^2 < v(j) + atv(j)^2 + 2b(x(j) + v(j)t + \frac{a}{2}t^2 - x(i) + v(i)t + \frac{a}{2}t^2) \wedge v(i) + atv(i), v(j) + atv(j) \geq 0)$$

$$\vee (x(i) + v(i)t + \frac{a}{2}t^2 > x(j) + v(j)t + \frac{a}{2}t^2 \wedge v(j) + atv(j)^2 < v(i) + atv(i)^2 + 2b(x(i) + v(i)t + \frac{a}{2}t^2 - x(j) + v(j)t + \frac{a}{2}t^2) \wedge v(i) + atv(i), v(j) + atv(j) \geq 0)) \text{ else}$$

$$(x(i) + v(i)t + \frac{a}{2}t^2 < x(j) + v(j)t + \frac{-b}{2}t^2 \wedge v(i) + atv(i)^2 < v(j) + -btv(j)^2 + 2b(x(j) + v(j)t + \frac{-b}{2}t^2 - x(i) + v(i)t + \frac{a}{2}t^2) \wedge v(i) + atv(i), v(j) + -btv(j) \geq 0)$$

$$\vee (x(i) + v(i)t + \frac{a}{2}t^2 > x(j) + v(j)t + \frac{-b}{2}t^2 \wedge v(j) + -btv(j)^2 < v(i) + atv(i)^2 + 2b(x(i) + v(i)t + \frac{a}{2}t^2 - x(j) + v(j)t + \frac{-b}{2}t^2) \wedge v(i) + atv(i), v(j) + -btv(j) \geq 0)) \text{ fi else}$$

$$\text{if}\, \forall i : C!\ \left(\left(x(i) > x(j) \to x(i) > x(j) + \frac{v(j)^2 - v(i)^2}{2b} + \left(\frac{a}{b} + 1\right)\left(\frac{a}{2}\varepsilon^2 + \varepsilon v(j)\right)\right)\right) \text{ then}$$

$$(x(i) + v(i)t + \frac{-b}{2}t^2 < x(j) + v(j)t + \frac{a}{2}t^2 \wedge v(i) + -btv(i)^2 < v(j) + atv(j)^2 + 2b(x(j) + v(j)t + \frac{a}{2}t^2 - x(i) + v(i)t + \frac{-b}{2}t^2) \wedge v(i) + -btv(i), v(j) + atv(j) \geq 0)$$

$$\vee (x(i) + v(i)t + \frac{a}{2}t^2 > x(j) + v(j)t + \frac{a}{2}t^2 \wedge v(j) + atv(j)^2 < v(i) + atv(i)^2 + 2b(x(i) + v(i)t + \frac{a}{2}t^2 - x(j) + v(j)t + \frac{a}{2}t^2) \wedge v(i) + atv(i), v(j) + atv(j) \geq 0)) \text{ else}$$

$$(x(i) + v(i)t + \frac{-b}{2}t^2 < x(j) + v(j)t + \frac{-b}{2}t^2 \wedge v(i) + -btv(i)^2 < v(j) + -btv(j)^2 + 2b(x(j) + v(j)t + \frac{-b}{2}t^2 - x(i) + v(i)t + \frac{-b}{2}t^2) \wedge v(i) + -btv(i), v(j) + -btv(j) \geq 0)$$

$$\vee (x(i) + v(i)t + \frac{a}{2}t^2 > x(j) + v(j)t + \frac{-b}{2}t^2 \wedge v(j) + -btv(j)^2 < v(i) + atv(i)^2 + 2b(x(i) + v(i)t + \frac{a}{2}t^2 - x(j) + v(j)t + \frac{-b}{2}t^2) \wedge v(i) + atv(i), v(j) + -btv(j) \geq 0)) \text{ fi fi}$$

$\mathcal{AS}_t\mathcal{M}_{i,j} \equiv$ similar instance

Figure 5: Qd$\mathcal{L}$ proof for collision freedom in distributed car control with acceleration.

if it only affects a finite subdomain of $\sigma(C)$, because the quantifier then corresponds to a finite set of classical differential equations. (The number of differential equations may still change dynamically over time, though, so that the quantified differential equation system *cannot* be replaced with an unquantified differential equation system in the QHP). For infinite $\sigma(C)$, the theorems carry over to schematic $\forall i\,{:}\,C\ f(i)' = \theta \,\&\, \chi$, which give an (infinite) set of disconnected classical differential equations. In all these cases, Picard-Lindelöf's theorem implies that the solution is unique, when terms are continuously differentiable (on the open domain where divisors are non-zero). For general infinite-dimensional differential equations see [3]. Qd$\mathcal{L}$ is also compatible with temporal operators that refer to intermediate states and nonterminating traces [19].

B Soundness and Relative Completeness Proof

In this section, we present a fully constructive proof of Theorem 1. First, we show that the QdL calculus is a sound axiomatization of QdL ([20]). Second, we prove that the QdL calculus is a complete axiomatization relative to quantified differential equations: every valid QdL formula can be derived in the QdL calculus from elementary properties of quantified differential equations (remainder of [20]).

In addition to the soundness proof, we present a relative completeness proofs for QdL. The basic structure follows that of our relative completeness proof for unquantified differential dynamic logic for fixed-dimensional static hybrid systems in previous work [17]. Here we generalize the proof to QdL. A fundamental difference to previous work is that states can be characterized trivially in fixed-dimensional static hybrid systems, but it is not obvious why a finite formula would be sufficient in varying dimensions. In (dynamic) distributed hybrid systems, we have to prove that there is a finite formula that can characterize and identify all states (see [20]). In fixed-dimensional static hybrid systems, states can be characterized and identified trivially by a fixed vector of real numbers for each system variable. In QdL, instead, states are full first-order structures with interpretations of functions for all function symbols and the ability to characterize semantic states in logic is no longer obvious. States are no longer assignments of real numbers to a finite number of variables. In QdL, states are first-order interpretations of function symbols.

As a basis for the relative completeness proof, we define FOQD as the *first-order logic of quantified differential equations*, i.e., first-order real arithmetic augmented with formulas expressing properties of quantified differential equations, that is, QdL formulas of the form $[\forall i\!:\!C\ f(\vec{s})' = \theta \,\&\, \chi]F$. Dually, $\langle\forall i\!:\!C\ f(\vec{s})' = \theta \,\&\, \chi\rangle F$ is expressible as $\neg[\forall i\!:\!C\ f(\vec{s})' = \theta \,\&\, \chi]\neg F$. Now the relative completeness direction of Theorem 1 corresponds to proving that for every valid QdL formula, there is a finite set of valid FOQD-formulas from which it can be derived in the QdL calculus. See Section 7 for a road map of the proof. We give the full proof below.

Natural numbers are definable in FOQD by a simple corollary to a previous result [17, Theorem 2]. Thus, we allow quantifiers over natural numbers like $\forall x : \mathbb{N}\ \phi$ and $\exists x : \mathbb{N}\ \phi$ and over integers $\forall x : \mathbb{Z}\ \phi$ as abbreviations.

B.1 Soundness Proof

For one direction of the proof for Theorem 1, we have to show that the QdL calculus in Fig. 2 is, indeed, a sound axiomatization. An unsound calculus would not be very interesting, and, in particular, it would not be sound and complete relative to quantified differential equations.

Theorem 2 (Soundness) *The* QdL *calculus is sound: every* QdL *formula that can be proven is valid, i.e., true in all states.*

Proof: The calculus is sound if each rule instance is sound. Some of the rules of the QdL calculus are even *locally sound*, i.e., their conclusion is true at state σ if all its premisses are true in σ, which implies soundness. The proofs for the propositional rules, and regular rules [;],⟨;⟩,[∪],⟨∪⟩,[?],⟨?⟩ are as usual. We refer to previous work [17] for the soundness proofs for ∃r,∀l,∀r,∃l,i∃, which are slightly more involved.

i∀ i∀ is locally sound. Assume the premiss

$$\sigma \models \mathrm{QE}(\forall X, Y\ (\textsf{if}\ \vec{s} = \vec{t}\ \textsf{then}\ \Phi(X) \to \Psi(X)\ \textsf{else}\ \Phi(X) \to \Psi(Y)\ \textsf{fi}))$$

Since QE yields an equivalence, we can conclude

$$\sigma \models \forall X, Y\ (\textsf{if}\ \vec{s} = \vec{t}\ \textsf{then}\ \Phi(X) \to \Psi(X)\ \textsf{else}\ \Phi(X) \to \Psi(Y)\ \textsf{fi})$$

This is equivalent to $\sigma \models \textsf{if}\ \vec{s} = \vec{t}\ \textsf{then}\ \forall X\ (\Phi(X) \to \Psi(X))\ \textsf{else}\ \forall X, Y\ (\Phi(X) \to \Psi(Y))\ \textsf{fi}$, because fresh variables X, Y do not occur in $\vec{s}$ or $\vec{t}$. Then assume the antecedent of the conclusion is true, i.e., $\sigma \models \Phi(f(\vec{s}))$. We conclude that the succedent of the conclusion is true, $\sigma \models \Psi(f(\vec{t}))$, by choosing $\sigma[\![f(\vec{s})]\!]$ for X and $\sigma[\![f(\vec{t})]\!]$ for Y in the premiss. If $\sigma \models \neg(\vec{s} = \vec{t})$ then $\sigma \models \Psi(f(\vec{t}))$ follows directly from the premiss. If, otherwise, $\sigma \models \vec{s} = \vec{t}$, then $\sigma \models \Psi(f(\vec{t}))$ also follows, because the choice $\sigma[\![f(\vec{s})]\!]$ for X is identical to the choice $\sigma[\![f(\vec{t})]\!]$ for Y in the premiss. By admissibility of substitutions, any variables occurring in terms $\vec{s}$ and $\vec{t}$ are free at all occurrences of $f(\vec{s})$ and $f(\vec{t})$, hence their value is the same in all occurrences.

$\langle :=\rangle$ Rule $\langle :=\rangle$ is locally sound for injective $\forall i : C\ f(\vec{s}) := \theta$, which we abbreviate as $\mathcal{A}$. Injective $\mathcal{A}$ give a deterministic transition. Assume the premiss

$$\sigma \models \textsf{if}\ \exists i : C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}\ \textsf{then}\ \exists i : C\ (\vec{s} = \langle\mathcal{A}\rangle\vec{u} \wedge \phi(\theta))\ \textsf{else}\ \phi(f(\langle\mathcal{A}\rangle\vec{u}))\ \textsf{fi}$$

We have to show that $\sigma \models \phi(\langle\forall i : C\ f(\vec{s}) := \theta\rangle f(\vec{u}))$. First assume that, with a fresh variable z, $\phi(z)$ is a first-order formula without modalities or quantifiers. Let τ be the (unique) state with $(\sigma, \tau) \in \rho(\forall i : C\ f(\vec{s}) := \theta) = \rho(\mathcal{A})$. By renaming, we can assume the quantified variable i not to occur anywhere else than in $\mathcal{A}$. We write this occurrence constraint as $i \notin \vec{u}$ and $i \notin \phi(z)$.

- Suppose $\sigma \models \exists i : C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}$, then $\sigma \models \exists i : C\ (\vec{s} = \langle\mathcal{A}\rangle\vec{u} \wedge \phi(\theta))$ by premiss. That is equivalent to: there is an $e \in \sigma(C)$ with $\sigma_i^e \models \vec{s} = \langle\mathcal{A}\rangle\vec{u} \wedge \phi(\theta)$. That means $\sigma_{i\,z}^{e\,d} \models \phi(z)$ for $d := \sigma_i^e[\![\theta]\!]$ by the substitution lemma. This is equivalent to $\sigma_z^d \models \phi(z)$, because $i \notin \phi(z)$, i.e., i does not occur in $\phi(z)$, so that its value is irrelevant. We want to show that $\sigma_z^d \models \phi(z)$ also holds for $d = \sigma[\![\langle\mathcal{A}\rangle f(\vec{u})]\!]$, because this implies $\sigma \models \phi(\langle\mathcal{A}\rangle f(\vec{u}))$ by the substitution lemma. Now

$$\sigma[\![\langle\mathcal{A}\rangle f(\vec{u})]\!] = \tau[\![f(\vec{u})]\!] = \tau(f)\big(\tau[\![\vec{u}]\!]\big) = \tau(f)\big(\sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!]\big) \overset{*}{=} \tau(f)\big(\sigma_i^e[\![\vec{s}]\!]\big) \overset{\rho(\mathcal{A})}{=} \sigma_i^e[\![\theta]\!] = d$$

Thus $\sigma \models \phi(\langle\mathcal{A}\rangle f(\vec{u}))$. The equality marked $*$ holds, because the premiss implies $\sigma_i^e \models \vec{s} = \langle\mathcal{A}\rangle\vec{u}$, which yields

$$\sigma_i^e[\![\vec{s}]\!] = \sigma_i^e[\![\langle\mathcal{A}\rangle\vec{u}]\!] \overset{i \notin \vec{u}}{=} \sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!]$$

- Suppose $\sigma \models \neg\exists i : C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}$, then $\sigma \models \phi(f(\langle\mathcal{A}\rangle\vec{u}))$ by premiss. Thus

$$\sigma_z^d \models \phi(z)$$

for $d := \sigma[\![f(\langle\mathcal{A}\rangle\vec{u})]\!]$ by the substitution lemma. We want to show that $\sigma_z^d \models \phi(z)$ also holds for $d = \sigma[\![\langle\mathcal{A}\rangle f(\vec{u})]\!]$, because this implies $\sigma \models \phi(\langle\mathcal{A}\rangle f(\vec{u}))$ by the substitution lemma. Now

$$\sigma[\![\langle\mathcal{A}\rangle f(\vec{u})]\!] = \tau[\![f(\vec{u})]\!] = \tau(f)\big(\tau[\![\vec{u}]\!]\big) \overset{*}{=} \sigma(f)\big(\tau[\![\vec{u}]\!]\big) = \sigma(f)\big(\sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!]\big) = \sigma[\![f(\langle\mathcal{A}\rangle\vec{u})]\!] = d$$

The equality marked $*$ holds, because—by assumption $\sigma \models \neg\exists i : C\ \vec{s} = \langle\mathcal{A}\rangle\vec{u}$—we know that for position $\tau[\![\vec{u}]\!] = \sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!]$ there is no $e \in \sigma(C)$ such that

$$\sigma_i^e[\![\vec{s}]\!] = \tau[\![\vec{u}]\!] = \sigma[\![\langle\mathcal{A}\rangle\vec{u}]\!] \overset{i \notin \vec{u}}{=} \sigma_i^e[\![\langle\mathcal{A}\rangle\vec{u}]\!]$$

Thus $\mathcal{A}$ has no effect on the interpretation of f at position $\tau[\![\vec{u}]\!]$ and σ and τ agree at that position.

In both cases, equivalence of premiss and conclusion can be established by following the equations and equivalences backwards, which also gives a proof for the dual rule $[:=]$. For the case where $\phi(z)$ contains modalities or quantifiers, the proof is accordingly using the substitution lemma and the fact that the interpretation of the symbols occurring in $\langle\mathcal{A}\rangle f(\vec{u})$ is not affected by the modalities and quantifiers in $\phi(z)$ (since all substitutions need to be admissible for Qd$\mathcal{L}$ rules to be applicable).

$skip$ Local soundness of rule $skip$ for injective quantified assignments $\forall i : C\ f(\vec{s}) := \theta$ is a simple consequence of the fact that a quantified assignment to f cannot affect the evaluation of another operator $\Upsilon \neq f$, but only its arguments (assuming admissible substitutions).

ex The soundness of axiom ex (i.e., validity of the conclusion) is a simple consequence of the fact that we have assumed finite support for the createdness flag $\mathsf{E}(\cdot)$ and that domains are infinite. That is, there are only finitely many $e \in \sigma(C)$ with $\sigma_i^e \models \mathsf{E}(i) = 1$, while domain $\sigma(C)$ is infinite. Consequently, in every state σ, there always is a choice e for i that has not been created yet ($\sigma_i^e \models \mathsf{E}(i) \neq 1$).

$\langle'\rangle$ Rule $\langle'\rangle$ is locally sound. Let $y_{\vec{s}}(t)$ be simultaneous solutions for the respective differential equations with symbolic initial values $f(\vec{s})$. Let $\langle\forall i : C\ \mathcal{S}(t)\rangle$ denote the quantified assignment

$$\langle\forall i : C\ f(\vec{s}) := y_{\vec{s}}(t)\rangle$$

Assume σ satisfies the premiss: $\sigma \models \exists t{\geq}0\, (\bar{\chi} \wedge \langle\forall i : C\ \mathcal{S}(t)\rangle\phi)$, with $\forall 0{\leq}\tilde{t}{\leq}t\ \langle\forall i : C\ \mathcal{S}(\tilde{t})\rangle\chi$ abbreviated as $\bar{\chi}$. By premiss, there is a real value $r \geq 0$ such that $\sigma_t^r \models \bar{\chi} \wedge \langle\forall i : C\ \mathcal{S}(t)\rangle\phi$. Abbreviate $\forall i : C\ f(\vec{s})' = \theta\ \&\ \chi$ by $\mathcal{D}$. We have to show that $\sigma \models \langle\mathcal{D}\rangle\phi$. Equivalently, we show $\sigma_t^r \models \langle\mathcal{D}\rangle\phi$, because t is a fresh variable that does not occur in $\mathcal{D}$ or ϕ. Let function $\varphi : [0, r] \to \mathcal{S}$ be defined such that $(\sigma, \varphi(\zeta)) \in \rho(\mathcal{S}(t))$ for all $\zeta \in [0, r]$. By premiss, $\varphi(0)$ is identical to σ and ϕ holds at $\varphi(r)$. Thus it only remains to be shown that φ respects the constraints for the flow function φ in the definition of the semantics of $\rho(\mathcal{D})$ in Section 4. In fact, φ obeys the continuity and differentiability properties required for well-definedness of time-derivatives by the corresponding properties of the solution $y_{\vec{s}}(t)$. Moreover, for any $e \in \sigma(C)$, $\varphi(\zeta)_i^e[\![f(\vec{s})]\!] = \sigma_{t\,i}^{\zeta\,e}[\![y_{\vec{s}}(t)]\!]$ has a derivative of value $\varphi(\zeta)_i^e[\![\theta]\!]$, because $y_{\vec{s}}$ is a solution of the quantified differential equation $\forall i : C\ f(\vec{s})' = \theta$ with corresponding initial values $\sigma(f(\vec{s}))$. Further, it can be shown that the evolution invariant region χ is respected along φ as follows: By premiss, $\sigma_t^r \models \bar{\chi}$ holds for the initial state σ_t^r, thus $\varphi(\zeta) \models \chi$ for all $\zeta \in [0, r]$. Combining these results, we can conclude that φ is a witness for $\sigma \models \langle\mathcal{D}\rangle\phi$.
The converse direction can be shown accordingly to prove equivalence and the dual rule $[']$ for quantified differential equations with unique solutions (see [20]). Without unique solutions, the rule is a little more complicated, but still works: all parameters of all parametric solutions will need to be quantified over in addition to time $t{\geq}0$.

$[:*]$ Rules $[:*]$,$\langle:*\rangle$ are locally sound by a simple consequence of the fact that arbitrary nondeterministic assignment of θ for any j of type C to n is the same as corresponding quantification over C. The semantics of $[\forall j : C\ n := \theta]$ then is equivalent to universal quantification, that of $\langle \forall j : C\ n := \theta \rangle$ is equivalent to existential quantification.

$\langle\rangle gen$ Rules $[]gen$,$\langle\rangle gen$,ind,con are sound (but not locally sound) by a variation of the usual proofs [10]. For $\langle\rangle gen$, let premiss $\phi \to \psi$ be valid. Let the antecedent be true in a state: $\sigma \models \langle\alpha\rangle\phi$, i.e., let $(\sigma, \tau) \in \rho(\alpha)$ with $\tau \models \phi$. Hence, the premiss implies $\tau \models \phi \to \psi$, thus $\tau \models \psi$, which implies $\sigma \models \langle\alpha\rangle\psi$. The proof for $[]gen$ is accordingly.

ind Let premiss $\phi \to [\alpha]\phi$ be valid and let the antecedent of the conclusion be true in σ, that is $\sigma \models \phi$. By premiss, $\tau \models \phi$ for all states τ with $(\sigma, \tau) \in \rho(\alpha)$. We thus conclude $\sigma \models \phi \to [\alpha^*]\phi$ by induction along the series of states reached from σ by repeating α.

con Assume the antecedent is valid and the premiss holds in σ. By premiss, we have

$$\tau \models v > 0 \wedge \varphi(v) \to \langle\alpha\rangle\varphi(v-1)$$

for all states τ. By antecedent, there is a $d \in \mathbb{R}$ such that $\sigma_v^d \models \varphi(v)$. Now, the proof is a well-founded induction on d. If $d \leq 0$, we directly have $\sigma \models \langle\alpha^*\rangle \exists v{\leq}0\, \varphi(v)$ for zero repetitions. Otherwise, if $d > 0$, we have, by premiss, that

$$\sigma_v^d \models v > 0 \wedge \varphi(v) \to \langle\alpha\rangle\varphi(v-1)$$

As $v > 0 \wedge \varphi(v)$ holds true at σ_v^d, we have for some τ with $(\sigma_v^d, \tau) \in \rho(\alpha)$ that $\tau \models \varphi(v-1)$. Thus, $\tau_v^{d-1} \models \varphi(v)$ satisfies the induction hypothesis for a smaller d and a reachable τ, because $(\sigma, \tau) \in \rho(\alpha)$ as v does not occur in α. The induction is well-founded, because d decreases by 1 up to the base case $d \leq 0$.

□

B.2 Characterizing Real Gödel Encodings

As the central device for constructing a FOQD formula that captures the effect of unboundedly many repetitive hybrid transitions and just uses finitely many real variables, we prove that a real version of Gödel encoding is definable in FOQD. That is, we give a FOQD formula that reversibly packs finite sequences of real values into a single real number.

Observe that a single differential equation system is *not* sufficient for defining these pairing functions as their solutions are differentiable, yet, as a consequence of Morayne's theorem [17, reference 43], there is no differentiable surjection $\mathbb{R} \to \mathbb{R}^2$, nor to any part of $\mathbb{R}^2$ of positive measure. We show that real sequences can be encoded nevertheless by chaining the effects of solutions of multiple differential equations and quantifiers.

Lemma 1 **($\mathbb{R}$-Gödel encoding)** *The formula at(Z, n, j, z), which holds iff Z is a real number that represents a Gödel encoding of a sequence of n real numbers with real value z at position j (for a position j with $1 \leq j \leq n$), is definable in FOQD. For a formula $\phi(z)$ we abbreviate $\exists z\, (\text{at}(Z, n, j, z) \wedge \phi(z))$ by $\phi(Z_j^{(n)})$.*

Proof: The proof is an immediate corollary to a result from previous work for a sublogic of Qd$\mathcal{L}$ [17].
□

B.3 First-order State Identification

The crucial step in the proof of Theorem 1 is the construction of Qd$\mathcal{L}$ (in)variants that are strong enough to characterize properties of repetition. In order to characterize QHP state transitions in Qd$\mathcal{L}$ (in)variants for the completeness proof, we first need to find formulas that characterize/identify states. For finite-dimensional systems of a fixed dimension n, states can simply be characterized completely by the values of all n real state variables. A particular state could be characterized uniquely by the formula $x = 2 \wedge y = 0.5 \wedge z = -0.382$, for example. As a trivial corollary to Lemma 1, states can then even be characterized uniquely by one real number when using the $\mathbb{R}$-Gödel encoding. For infinite-dimensional systems, systems with changing dimension, or systems with a dynamics that depends on evolving interpretations of function symbols $f(\vec{s})$, the situation is more difficult. After all, a state of Qd$\mathcal{L}$ is a full first-order structure with functions as interpretations of function symbols, and these interpretations can change from state to state. Furthermore, in order to navigate among states during the completeness proof, we need to be able to characterize the current first-order state, but also to recall a previously identified first-order state and express what holds true at this state.

We show that the first-order states reachable with QHP α from an initial state can be characterized uniquely by real numbers, which can thus be quantified over. Furthermore, we show that this correspondence can be axiomatized in FOQD. One key observation is that the first-order interpretations can change from state to state, but only according to the dynamics of the QHP Intuitively, the difference of any reachable first-order state to the initial state can be characterized by a finite list of differences to the initial state. Clearly this difference concerns only finitely many symbols occurring in α. It also concerns only finitely many positions of their interpreted functions, because actualist quantified assignments and actualist quantified differential equations only change the interpretation of finitely many function symbols at finitely many positions (actual quantified domains $C!$ occurring in actualist quantifiers of QHPs are finite). Note that it is crucial here that we have assumed the actual existence predicate $\mathsf{E}(i)$ to have finite support.

Lemma 2 (State identification) *Let Σ_b be a finite set of function symbols containing $\mathsf{E}(\cdot)$. The operators $\downarrow$ and $@$, which identify and recall states reachable by QHPs, are definable in FOQD such that:*

1. *For every QHP α with $\mathrm{BV}(\alpha) \subseteq \Sigma_b$, every variable $\mathfrak{I} \notin \Sigma_b$ of sort $\mathbb{R}$, and every state σ, the formula $\downarrow \mathfrak{I}$ is true in at most one of the states reachable by α from σ. That is, there is at most one state ι such that $(\sigma, \iota) \in \rho(\alpha)$ and $\iota \models \downarrow \mathfrak{I}$.*
2. *For every QHP α with $\mathrm{BV}(\alpha) \subseteq \Sigma_b$, every variable $\mathfrak{I} \notin \Sigma_b$ of sort $\mathbb{R}$, every formula ϕ, and every state σ, the formula $@\,\mathfrak{I}\,\phi$ is true in any state reachable by α from σ if and only if ϕ is true in the (unique) state that is reachable by α from σ in which $\downarrow \mathfrak{I}$ holds (provided such a state is reachable at all, otherwise the truth-value of $@\,\mathfrak{I}\,\phi$ is arbitrary). That is, suppose there is a state ι such that $(\sigma, \iota) \in \rho(\alpha)$ and $\iota \models \downarrow \mathfrak{I}$ (thus, by case 1, ι is unique with that property). Then for any state τ with $(\sigma, \tau) \in \rho(\alpha)$, it is the case that $\tau \models @\,\mathfrak{I}\,\phi$ if and only if $\iota \models \phi$. If, on the contrary, there is no state ι with $(\sigma, \iota) \in \rho(\alpha)$ and $\iota \models \downarrow \mathfrak{I}$, then this lemma makes no statement concerning the truth of formula $@\,\mathfrak{I}\,\phi$ at any state τ.*

Proof: The formulas $\downarrow \mathfrak{I}$ and $@\,\mathfrak{I}\,\phi$ are like the *here* and *at* operators of hybrid-nominal logic. They can be characterized by the following FOQD formulas:

$$\theta = is_f(\mathfrak{I}, \vec{o}) \equiv \text{if}\, \exists s : \mathbb{N}\ (s < m \land X_s^{(m)} = \vec{o})\ \text{then}\, \exists s : \mathbb{N}\ (s < m \land X_s^{(m)} = \vec{o} \land \theta = Y_s^{(m)})\ \text{else}\, \theta = f(\vec{o})\ \text{fi}$$

where $\mathfrak{I}$ is split into the following abbreviations $m := \mathfrak{I}_i^{(d)}{}_1^{(3)}, X := \mathfrak{I}_i^{(d)}{}_2^{(3)}, Y := \mathfrak{I}_i^{(d)}{}_3^{(3)}$ further d is the number of symbols in Σ_b and i is the index of f in Σ_b

$$\downarrow \mathfrak{I} \equiv \bigwedge_{f \in \Sigma_b} \forall \vec{o} : S_f\ f(\vec{o}) = is_f(\mathfrak{I}, \vec{o}) \qquad \text{where } S_f \text{ is the sort of the arguments of } f$$

$$@\,\mathfrak{I}\,\phi \equiv \langle \forall i : C\ \forall u : \mathbb{R}\ f(i)' = u \rangle (\phi \land \downarrow \mathfrak{I})$$

For defining $\downarrow \mathfrak{I}$ and $@\,\mathfrak{I}\,\phi$, we use an auxiliary function $is_f(\mathfrak{I}, \vec{o})$ to improve readability. The definitions do not need recursion, so that we can consider occurrences of the defined notations as syntactic abbreviations for quantified variables satisfying the respective definitions (like for Lemma 1).

The function symbol $is_f(\mathfrak{I}, \vec{o})$ gives the value (θ) of function f at position $\vec{o}$ at the state characterized by the real number denoted by $\mathfrak{I}$. It can be defined easily using the real pairing function from Lemma 1. The basic idea is to understand $\mathfrak{I}$ via the real pairing function as a list of length m of position/value pairs ($X_s^{(m)}/Y_s^{(m)}$), which characterize changes to the value $f(\vec{o})$ for each of the finitely many function symbols $f \in \Sigma_b$. Using an arbitrary but fixed ordering, these function symbols f are identified with their index d in Σ_b. The most important insight for the proof is that, for every state reachable by α from σ, the list of changes of f compared to $f(\vec{o})$ at σ is always finite after finitely many transitions of quantified state change with finite support (see end of Section 5). Consequently, the list of changes can always be encoded by one (finite) real number according to Lemma 1. Using this auxiliary function, we characterize cases 1 and 2:

Case 1: The characterization for $\downarrow \mathfrak{I}$ is defined as a conjunction over all relevant function symbols $f \in \Sigma_b$ asserting that the value $f(\vec{o})$ of f at each position $\vec{o}$ of the sort S_f of f is identical to the corresponding value $is_f(\mathfrak{I}, \vec{o})$ characterized by $\mathfrak{I}$.

Case 2: The characterization for $@\,\mathfrak{I}\,\phi$ uses a quantified differential equation with a variable u that only occurs on the right hand side and thus changes f at all positions i with an arbitrary slope u. The $@\,\mathfrak{I}\,\phi$ characterization then checks if the appropriate state characterized by $\mathfrak{I}$ has been reached using $\downarrow \mathfrak{I}$ and further expresses that ϕ holds at this state. By case 1, we know that $\downarrow \mathfrak{I}$ holds in at most one of the states reachable by α from σ. In the quantified differential equation system for $@\,\mathfrak{I}\,\phi$, the second quantified variable u amounts to nondeterministically specifying a slope u for each $f(i)$. Unlike i, quantified variable u only occurs on the right hand side of the quantified differential equation. Consequently, the semantics (case 2 of the transition relation $\rho(\alpha)$ defined in Section 4) defines the states corresponding to *all choices* for u to be reachable. These respective choices for u include the choice that leads to the state characterized by $\downarrow \mathfrak{I}$, e.g., by choosing slope $u := is_f(\mathfrak{I}, i) - f(i)$ for each i and evolving for 1 time unit. To simplify notation, we define $@\,\mathfrak{I}\,\phi$ only for $\Sigma_b = \{f\}$. The construction is repeated accordingly (by nesting modalities) for each $f \in \Sigma_b$, which are finitely many. The createdness flag $\mathsf{E}(\cdot)$ needs to be part of Σ_b for object creation to be taken care of. $\square$

The rest of the proof follows similar principles to those in [17]. The full proof is beyond the scope of this report.